Refine Search

Search Results -

Term	Documents
(1 AND 11 AND 4 AND 6 AND 3).USPT.	2
(L1 AND L11 AND L6 AND L3 AND L4).USPT.	2

Database:
US Pre-Grant Publication Full-Text Database
US Patents Full-Text Database
US OCR Full-Text Database
EPO Abstracts Database
JPO Abstracts Database
Derwent World Patents Index
IBM Technical Disclosure Bulletins

Search: L17

Refine Search

Recall Text | Clear | Interrupt

Search History

DATE: Wednesday, May 12, 2004 Printable Copy Create Case

Set Name side by side	Query	Hit Count	Set Name result set
DB=USPT; PLUR=YES; OP=ADJ			
L17	l1 and l11 and l6 and l3 and l4	2	L17
L16	l6 and L15	2	L16
L15	l4 and L14	2	L15
L14	l9 and L13	3	L14
L13	l1 and L11	8	L13
L12	l10 and L11	0	L12
L11	(modif$ or chang$) near3 l6	3791	L11
L10	l8 and L9	9	L10
L9	migrat$	110504	L9
L8	l1 and l3 and l2 and l4 and l5 and l6 and L7	9	L8
L7	run time$1	15836	L7
L6	priorit$	146724	L6
L5	optimiz$	234468	L5

h e b b cg b e e ch

L4	RAID	3598	L4
L3	data near2 transfer$	106129	L3
L2	throughput or utilization rate$1	80401	L2
L1	hierarchical storage$1	430	L1

END OF SEARCH HISTORY

h e b b cg b e e ch

Refine Search

Search Results -

Term	Documents
"6321252"	1
6321252S	0
"6321252".PN..USPT.	1
(6321252.PN.).USPT.	1

Database:
US Pre-Grant Publication Full-Text Database
US Patents Full-Text Database
US OCR Full-Text Database
EPO Abstracts Database
JPO Abstracts Database
Derwent World Patents Index
IBM Technical Disclosure Bulletins

Search: L14

Refine Search

Recall Text | Clear | Interrupt

Search History

DATE: Wednesday, May 12, 2004 Printable Copy Create Case

Set Name side by side	Query	Hit Count	Set Name result set
	DB=USPT; PLUR=YES; OP=ADJ		
L14	6321252.pn.	1	L14
L13	6499026.pn.	1	L13
L12	6731625.pn.	1	L12
L11	l9 and L10	16	L11
L10	priorit$	146724	L10
L9	l7 and L8	18	L9
L8	optimiz$	234468	L8
L7	l5 and L6	20	L7
L6	RAID	3598	L6
L5	l2 and l3 and L4	55	L5
L4	throughput or utilization rate	80394	L4

h e b b cg b e e ch

L3	data near2 transfer$	106129	L3
L2	hierarchical storage	430	L2
L1	hierarchical storage near1 device	10	L1

END OF SEARCH HISTORY

Refine Search

Search Results -

Term	Documents
(9 AND 10).USPT.	16
(L9 AND L10).USPT.	16

Database:
US Pre-Grant Publication Full-Text Database
US Patents Full-Text Database
US OCR Full-Text Database
EPO Abstracts Database
JPO Abstracts Database
Derwent World Patents Index
IBM Technical Disclosure Bulletins

Search: L11

Refine Search

Recall Text | Clear | Interrupt

Search History

DATE: Wednesday, May 12, 2004 Printable Copy Create Case

Set Name side by side	Query	Hit Count	Set Name result set
DB=USPT; PLUR=YES; OP=ADJ			
L11	l9 and L10	16	L11
L10	priorit$	146724	L10
L9	l7 and L8	18	L9
L8	optimiz$	234468	L8
L7	l5 and L6	20	L7
L6	RAID	3598	L6
L5	l2 and l3 and L4	55	L5
L4	throughput or utilization rate	80394	L4
L3	data near2 transfer$	106129	L3
L2	hierarchical storage	430	L2
L1	hierarchical storage near1 device	10	L1

END OF SEARCH HISTORY

h e b b cg b e e ch

Hit List

Clear | Generate Collection | Print | Fwd Refs | Bkwd Refs | Generate OACS

Search Results - Record(s) 1 through 16 of 16 returned.

☐ 1. Document ID: US 6625750 B1

L11: Entry 1 of 16 File: USPT Sep 23, 2003

US-PAT-NO: 6625750
DOCUMENT-IDENTIFIER: US 6625750 B1

TITLE: Hardware and software failover services for a file server

DATE-ISSUED: September 23, 2003

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Duso; Wayne W.	Shrewsbury	MA		
Kuczynski; Leslie E.	Belmont	MA		
Forecast; John	Newton	MA		
Gupta; Uday	Westford	MA		
Vahalia; Uresh K	Newton	MA		
Ting; Dennis P. J.	Groton	MA		

ASSIGNEE-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY	TYPE CODE
EMC Corporation	Hopkinton	MA			02

APPL-NO: 09/ 440756 [PALM]
DATE FILED: November 16, 1999

INT-CL: [07] H02 H 3/05

US-CL-ISSUED: 714/11; 714/4, 714/13
US-CL-CURRENT: 714/11; 714/13, 714/4

FIELD-OF-SEARCH: 714/4-5, 714/9-15

PRIOR-ART-DISCLOSED:

U.S. PATENT DOCUMENTS

PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL
4577272	March 1986	Ballew et al.	364/200
5051887	September 1991	Berger et al.	364/200

h e b b g e e e f e ef b e

First Hit Fwd Refs

End of Result Set

Generate Collection Print

L21: Entry 1 of 1 File: USPT Dec 5, 2000

DOCUMENT-IDENTIFIER: US 6157963 A
TITLE: System controller with plurality of memory queues for prioritized scheduling of I/O requests from priority assigned clients

Abstract Text (1):
A system for globally prioritizing and scheduling I/O requests from a plurality of storage users or clients to one or more storage objects. The system comprises a storage controller configured to receive I/O requests from the client workstations and prioritize and schedule those I/O requests in accordance with a scheduling algorithm. Specifically, the storage controller receives I/O requests from the storage users and places the I/O requests in memory queues associated with the particular storage users. The storage controller then selects the I/O requests from the various memory queues based on the scheduling algorithm.

Brief Summary Text (2):
The present invention relates generally to methods and apparatus for scheduling I/O requests and specifically to a system for scheduling I/O requests from workstations to one or more particular storage objects.

Brief Summary Text (3):
A large number of distributed network environments typically comprise a number of client workstations networked to one or more centralized storage locations, such as, file servers or disk array subsystems. File servers typically comprise PC or RISC-processor workstations equipped with high storage capacity disk drives. Similarly, disk array subsystems typically comprise one or more disk arrays, for example, RAID arrays, and one or more array controllers attached thereto. The client workstations may be connected to the file servers and/or disk array controllers either through an I/O interconnect, such as SCSI bus, Fibre Channel or other similar I/O connection or the workstation may be connected through a network environment, such as Ethernet, Token Ring, FDDI/CDDI or the like.

Brief Summary Text (5):
In many business environments, certain users may be performing tasks which require quicker response times than tasks initiated by other users, or it may be desirable for certain users to have a priority higher than other users. In addition, certain data types requiring higher access priorities (e.g., a virtual memory swap volume) may be dedicated to specific disk drives or disk volumes, thus requiring all I/O requests directed to those specific volumes to be processed first. Unfortunately, however, the current disk storage interfaces (i.e., file server processors and/or array controllers) do not provide means for the global scheduling of I/O requests to one or more file servers or disk drive arrays, and therefore only can process I/O requests on a first come, first served basis. In addition, the disk storage interfaces currently known in the art do not have means for resolving contention issues between users, so if one user submits a plurality of I/O requests at one time, requests from other users can become blocked behind the I/O requests of that one user. Thus, higher priority I/O requests can be starved of resources and undesirably delayed.

Brief Summary Text (7):
Accordingly, it is an advantage of the present invention to provide methods and apparatus for scheduling I/O requests to one or more disk drives or arrays which overcome the shortcomings of the prior art.

Brief Summary Text (9):
Yet another advantage of the present invention is that it may use any one of a number of different prioritization and scheduling schemes currently known in the art or hereinafter developed.

Brief Summary Text (10):
Yet another advantage of the present invention is that the prioritization and scheduling scheme may be configured to change priority values assigned to memory queues, clients, storage objects, or the like to ensure that certain I/O requests placed in lower priority memory queues or originating from lower priority clients are not starved of processor resources.

Brief Summary Text (12):
The above and other advantages of the present invention are carried out in one form by a storage controller configured to prioritize and schedule I/O requests from a plurality of clients to at least one storage object. The storage controller suitably comprises a processing means for processing the I/O requests, a first interface means for sending and receiving data between the clients and the processing means, and a second interface means for sending and receiving data between the processing means and the storage objects.

Brief Summary Text (13):
Preferably, the processing means is configured to receive I/O requests from the clients via the first interface means, prioritize and schedule the I/O requests in accordance with a priority algorithm, and conduct an I/O operation with the storage objects via the second interface means for an I/O request having a preselected priority. After the I/O operation is complete, the processor communicates the results back to the client that initiated the I/O request.

Drawing Description Text (8):
FIG. 6 is a table illustrating a number of scheduling sequences preferred by one embodiment of the present invention.

Detailed Description Text (2):
The present invention is a method and apparatus for receiving, prioritizing, scheduling and processing I/O requests from a plurality of storage users or clients to one or more storage objects, where each storage object preferably is under the control of a central storage controller. For example, a suitable storage controller may be a block-access type storage device such as a disk controller or RAID storage controller. In such a case, the storage users or clients are host systems connected to the storage controller via an I/O interconnect such as a SCSI bus, Fibre Channel, or other suitable communication interface. The storage objects would typically be SCSI logical units, for example. In an alternative embodiment, the storage controller may be a network-based file server. In such a case, the storage users would be network-attached file access clients using network protocols such as Sun Microsystem's Network File System (NFS), Microsoft's Server Message Block (SMB) protocol or the like. The network attachment could be based on an Ethernet, FDDI/CDDI, Token Ring, or any suitable communication backbone. The storage objects typically would be the file systems exported by the file server to its clients. In any event, the methods and apparatus disclosed and claimed herein may be embodied in any suitable control device with any suitable communication scheme for conveying requests from the storage users to the storage controller. Accordingly, the present invention is not limited to the specific embodiments illustrated in the figures or disclosed herein.

Detailed Description Text (6):
As mentioned briefly above, storage controller 16 may comprise any suitable storage controller. For example, storage controller 16 may be a file server controller, a disk drive controller, or a RAID controller.

Detailed Description Text (10):
Similarly, storage object interface 26 is configured to communicate I/O requests from processor 24 to storage object(s) 20 via storage object connection 18. In accordance with a preferred embodiment of the present invention, storage object connection 18 comprises a SCSI bus architecture, and storage object interface 26 suitably comprises a standard SCSI interface. However, as with storage channel/network connection 14 and storage user interface 22, any suitable connection means may be utilized. For example, disk drive or RAID storage controllers likely will use high speed I/O interconnects, such as SCSI, Fibre Channel or the like, while file servers may communicate with a variety of storage objects, either directly or via a network connection such as Ethernet, FDDI/CDDI, Token Ring or the like.

Detailed Description Text (11):
Processor 24 may comprise any suitable computer processor, such as a microprocessor, microcontroller, or the like, which will prioritize and schedule requests from storage users for storage objects. In addition, while a preferred embodiment of the present invention is described herein with reference to a suitable processor, one skilled in the art will appreciate that the processing functions may be performed by other logic devices known in the art, such as ASIC or other hardwired logic devices.

Detailed Description Text (12):
Moreover, while the specific control of storage user interface 22 and storage object interface 26 is not specifically disclosed herein, one skilled in the art will appreciate that any suitable interface control device may be used. For example, each interface 22, 26 may include their own I/O processors, or processor 24 may control the interfaces. That is processor 24 may be a multitasking processor which handles the storage user I/O's, the storage object I/O's and the prioritization and scheduling tasks of the storage controller.

Detailed Description Text (15):
After placing the I/O requests in the particular memory queues 32, processor 24 then retrieves I/O requests from memory queues 32 in accordance with a predetermined scheduling algorithm (e.g., highest priority first) (Block 54). Processor 24 then processes the I/O operation for the selected I/O request with the appropriate storage object(s) 20, and returns the results of the I/O operation back to the requesting storage user 12 (Block 56).

Detailed Description Text (16):
In accordance with one aspect of the present invention, after processor 24 executes an I/O operation from one of the memory queues 32, processor 24 preferably changes the priority value assigned to that memory queue 32 (Block 58). By changing the priority value of memory queue 32 after an I/O request has been processed from that queue, processor 24, and in particular the prioritizing and scheduling algorithm used by processor 24, ensures that I/O requests placed in queues with lower initial priorities eventually get processed. If the priority values are not adjusted, one or a small number of relatively high priority queues may consume substantially all of the processing resources, preventing I/O requests in other lower priority queues from being processed. In addition, as discussed in more detail below, processor 24 periodically may perform a decay function on all the memory queues, changing the priority of all the queues in accordance with a predetermined decay function algorithm.

Detailed Description Text (18):

While the above discussion describes the operation of processor 24 in somewhat of a sequential manner, one skilled in the art will appreciate that processor 24 may perform many tasks virtually simultaneously, and may perform some of the tasks in different orders. For example, processor 24 may receive, prioritize and place an I/O request in memory 28 while it is also processing an I/O request from another storage user 12 or from the same storage user having multiple requests. In any event, the operation of processor 24 is not limited to the specific operating order and method described herein.

Detailed Description Text (20):
Referring still to FIG. 2, one particular prioritizing scheme in accordance with the present invention will be described. In particular, each storage user 12 preferably is assigned a fixed relative priority value. In accordance with this example, Fi will be used to represent the fixed priority value associated with a particular storage user i. Furthermore, a dynamic priority value Pi is also tracked for each storage user i. Finally a numeric value UPi is maintained for each storage user i. This value represents a "usage penalty" that is incurred each time an I/O request for a storage user i is processed by storage controller 16. For Pi, Fi and UPi, numerically larger values correspond to less favorable scheduling opportunities for storage user i. The dynamic priority value Pi for a given storage user i is calculated using the formula Pi=Fi+UPi.

Detailed Description Text (21):
When processor 24, and in particular the scheduler of processor 24 is selecting the memory queue 32 from which to extract the next idle request for processing, processor 24 preferably chooses the memory queue 32 associated with the storage user i whose dynamic priority value Pi is numerically lowest. To avoid confusion, numerically lower/lowest priority values are "better/best" and numerically higher/highest priority values are "worse/worst".

Detailed Description Text (22):
To further clarify the calculation of priority values on a per user basis, two other considerations will be addressed. First, as discussed above, each time a storage user's memory queue is selected for service, processor 24 adjusts the usage penalty UPi for that user so that the storage user's dynamic priority Pi is also adjusted. For example, after processor 24 selects an I/O request from a memory queue 32 associated with a storage user i, processor 24 adjusts the usage penalty UPi as follows: UPi=UPi+1. Accordingly, since the dynamic priority value Pi for user i is Pi=Fi+UPi, the dynamic priority value Pi is also adjusted when the usage potential is adjusted. This process serves to worsen the priority value for storage user i during the next scheduling interval, and thus penalizes the user and prevents it from monopolizing the storage controller resources.

Detailed Description Text (26):
Referring now to FIG. 6, a specific example illustrating the prioritization algorithm discussed above will now be described. The present example disclosed herein assumes the existence of three storage users having the following fixed priority values: F1=8, F2=9 and F3=10. Also, in accordance with this example, a static load factor of L=3 is assumed. The table illustrated in FIG. 6 tracks the dynamic priority values P for each storage user. The table also identifies the best dynamic priority value Fi at each scheduling interval, thus indicating the storage user whose memory queue is selected for service. Note also that some entries in the table shown in FIG. 6 are illustrated as "decay" intervals. These decay intervals are times at which the processor/scheduler recognizes the need to apply the decay factor to the usage penalty values UPi for each storage user.

Detailed Description Text (31):
As mentioned above, at certain sequences, the processor executes a decay factor algorithm. In the illustrated example, the first decay factor algorithm occurs between sequence six and sequence seven. As mentioned previously, the decay factor

algorithm for this particular example is UPi=UPi*L/(L+1). Thus, the usage penalty for storage user 1 will be altered as follows: UP1=3*3/4=2.3. Similarly, the usage penalty for storage user 2 is: 2*3/4=1.5 and the usage penalty for storage user 3 is: UP3=1*3/4=0.8. After the decay function has been applied, the processor proceeds again with scheduling the I/O requests as before.

Detailed Description Text (32):
In accordance with another aspect of the present invention, any number of scheduling algorithms may be utilized to select the I/O requests from memory 28, and in particular memory queues 32. For example, processor 24 may utilize the round-robin or FIFO scheduling algorithms, or a scheduling algorithm similar to that utilized by the UNIX operating system may be used. The scheduling algorithm described in the example above is a variation of the UNIX scheduling scheme. For a more detailed discussion of the UNIX operating systems scheduling scheme see, for example, "The Design and Implementation of the 4.3 BSD UNIX Operating System"; S Leffler, M. McKusick, M. Karel, and J. Quarterman; Addison-Wesley, 1989; ISBN 0-201-06196-1, which is incorporated herein by reference.

Detailed Description Text (33):
In addition, while the above discussion describes a prioritization scheme in which I/O requests are prioritized in accordance with a priority value assigned to storage users 12, any number of priority schemes may be utilized. For example, I/O requests may be prioritized by storage user priority, storage object priority, job priority or a combination of all of these factors. In the case of storage object priority, each storage object will be associated with a particular memory queue, and will be assigned fixed priority, usage penalty and dynamic priority values as discussed above. Similarly, as one skilled in the art will appreciate, the user of the system may change the priority schemes and/or algorithms used by the system, and the user may change the priorities assigned to the storage users and/or storage objects, both prior to I/O processing and dynamically during I/O processing. In accordance with this aspect of the invention, the user can change the storage user and/or storage object priorities manually via a user interface, or the priority values may be downloaded to the system and in particular to the storage controller by tape, disk or CD, or through a network environment. In any event, the system is dynamically configurable.

Detailed Description Text (34):
In accordance with another embodiment of the present invention, and referring to FIGS. 3 and 5, a plurality of prioritizing schemes may be used to prioritize and schedule the I/O requests. In accordance with this embodiment of the invention and as illustrated in FIG. 5, I/O requests from storage users 12 are first prioritized by storage user and then by storage object 20. More particularly, one or more of storage users 12 first send I/O requests to processor 24 (Block 60). Processor 24 receives the I/O requests and determines, using a first prioritization factor, in which one of memory queues 32 in a first storage area 30-1 it should place the I/O request (see FIG. 3). In accordance with a preferred embodiment of the invention, the first prioritization factor is storage user priority, and processor 24 places the I/O requests in memory queues 32 in first storage area 30-1 associated with the particular storage user 12 (Block 62). As discussed previously, particular memory queues 32 are configured to receive I/O requests from particular storage users 12 and are assigned priority values which are periodically adjusted during processing.

Detailed Description Text (35):
After placing one or more I/O requests in the particular memory queues 32 in first storage area 30-1, processor 24 then retrieves I/O requests from the memory queues 32 in accordance with a predetermined scheduling algorithm (e.g., highest priority first) (Block 64), and determines, using a second prioritization factor, in which one of memory queues 32 in a second storage area 30-2 it should place the I/O requests. In accordance with the illustrated embodiment of the present invention,

the second prioritization factor is storage object priority, and processor 24 places I/O requests in memory queues 32 associated with the particular storage objects 20 (Block 66).

Detailed Description Text (36):
After processor 24 removes an I/O request from one of the memory queues in first storage area 30-1, processor 24 preferably changes the priority value assigned to that memory queue 32 (Block 68).

Detailed Description Text (37):
Next, processor 24 preferably retrieves an I/O request from memory queues 32 in second storage area 30-2 in accordance with a predetermined scheduling algorithm (Block 70), and processes the I/O operation for the selected I/O request with the appropriate storage object(s) 20 (Block 72). The processor then returns the results of the I/O operation back to the requesting client 12. As with the memory queues 32 in first storage area 30-1, processor 24 preferably changes the priority value assigned to memory queue 32 in second storage area 30-2 from which the last I/O request was taken (Block 74). In addition, as mentioned above, processor 24 periodically may perform a decay function on all the queues, changing the priority of all the memory queues in accordance with a predetermined decay function algorithm.

Detailed Description Text (38):
After processor 24 adjusts the priority of memory queue 32 in second storage area 30-2, processor 24 returns to block 64 to select another I/O request from one of the memory queues 32 in accordance with the prioritization and scheduling algorithm.

Detailed Description Text (39):
Again, as discussed above with reference to FIG. 4, the operation of processor 24 is not necessarily limited to the specific order of operation described herein, which is merely an example of a preferred embodiment. For example, processor 24 may prioritize and store several I/O requests in first storage area 30-1 before it processes any I/O operations for I/O requests in second storage are 30-2. Similarly, processor 24 may change the priority of memory queues 32 in first storage area 30-1 and second storage area 30-2 at any time throughout the process. In any event, the present invention is not limited to the illustrated embodiment.

Detailed Description Text (40):
In conclusion, the present invention provides a novel method and apparatus for prioritizing and scheduling I/O requests from one or more clients to one or more storage objects. While a detailed description of presently preferred embodiments of the invention have been given above, various alternatives, modifications, and equivalents will be apparent to those skilled in the art. For example, any suitable storage controller may be utilized, and any number of different prioritization and scheduling schemes and algorithms may be used without varying from the spirit of the invention. Therefore, the above description should not be taken as limiting the scope of the invention which is defined by the appended claims.

CLAIMS:

1. A storage controller configured to prioritize and schedule I/O requests from a plurality of clients to at least one storage object, comprising:

processing means for processing I/O requests from said plurality of clients in a prioritized order;

first interface means for sending and receiving data between said clients and said processing means; and

second interface means for sending and receiving data between said processing means and said at least one storage object;

said processing means being configured to receive I/O requests from said clients via said first interface means, prioritize and schedule said I/O requests based on a priority assigned to each of said clients, conduct an I/O operation with said at least one storage object via said second interface means for an I/O request from a client having a highest priority, communicate results from said I/O operation back to the client which initiated the I/O request processed by the I/O operation, and change the priority assigned to the client which initiated the I/O request processed by the I/O operation.

4. The storage controller as recited in claim 1 wherein said processing means periodically changes said priority assigned to each one of said plurality of clients in accordance with a priority decay algorithm.

6. A storage controller configured to prioritize and schedule I/O requests from one or more clients to a plurality of storage objects, comprising:

processing means for processing I/O requests from said one or more clients in a prioritized order;

first interface means for sending and receiving data between said one or more clients and said processing means; and

second interface means for sending and receiving data between said processing means and said plurality of storage object;

said processing means being configured to receive I/O requests from said one or more clients via said first interface means, prioritize and schedule said I/O requests based on a priority assigned to each of said storage objects, conduct an I/O operation with one or more of said storage objects via said second interface means for an I/O request directed to a storage object having a highest priority, communicate results from said I/O operation back to the client which initiated the I/O request processed by the I/O operation, and change the priority assigned to the one or more storage objects with which the I/O operation was conducted.

7. A storage controller configured to prioritize and schedule I/O requests from a plurality of clients to at least one storage object, comprising:

a memory having a plurality of memory queues;

processing means for processing I/O requests from said plurality of clients in a prioritized order;

first interface means for sending and receiving data between said clients and said processing means; and

second interface means for sending and receiving data between said processing means and said at least one storage object;

said processing means being configured to receive I/O requests from said clients via said first interface means, place each one of said I/O requests into one of said plurality of memory queues based on a priority of said I/O request, remove an I/O request having a selected priority from one of said plurality of memory queues, conduct an I/O operation with said at least one storage object via said second interface means for the removed I/O request, and communicate results from said I/O operation back to the client which initiated the I/O request.

13. A method for prioritizing and scheduling I/O requests from a plurality of

clients to one or more storage objects, comprising the steps of:

receiving via a first communication interface I/O requests from said plurality of clients;

prioritizing and scheduling said I/O requests based on a priority assigned to each of said clients;

conducting an I/O operation with said one or more storage objects for an I/O request from a client having a highest priority;

communicating results from said I/O operation back to the client which initiated the I/O request processed by the I/O operation; and

changing the priority assigned to the client which initiated the I/O request processed by the I/O operation.

14. The method as recited in claim 13 further comprising the step of periodically changing said priority assigned to each one of said plurality of clients in accordance with a priority decay algorithm.

15. A system for scheduling I/O requests to at least one storage object from a plurality of clients, comprising:

a processor;

a client interface configured to facilitate communications between said plurality of clients and said processor;

a storage object interface configured to facilitate communications between said processor and said at least one storage object; and

a memory having a plurality of memory queues;

wherein one or more of said plurality of clients send I/O requests to said processor via said client interface, said processor receives said I/O requests and places each one of said I/O requests in one of said plurality of memory queues based on a priority of said I/O requests, and at an approprate time said processor removes an I/O request having a selected priority value from one of said plurality of memory queues and conducts an I/O operation with said at least one storage object.

22. The system as recited in claim 20 wherein said processor changes said priority assigned to a memory queue after said processor conducts an I/O operation for an I/O request from said memory queue.

23. The system as recited in claim 20 wherein said processor periodically changes said priority assigned to each one of said plurality of memory queues in accordance with a priority decay algorithm.

27. The system as recited in claim 15 wherein said system is configured to schedule I/O requests directed to a disk array.

28. The system as recited in claim 15 wherein said system is configured to schedule I/O requests directed to a file server.

First Hit Fwd Refs
End of Result Set

Generate Collection Print

L33: Entry 3 of 3 File: USPT Jan 4, 2000

DOCUMENT-IDENTIFIER: US 6012123 A
** See image for Certificate of Correction **
TITLE: External I/O controller system for an independent access parity disk array

Abstract Text (1):
An external I/O controller system between a host system and a Redundant Array of Independent Disks (RAID) array, having a processor complex for coordinating RAID parity write operations to a member disk in a RAID array, and a cache memory complex that asynchronously executes the RAID parity write operation independent of the processor complex and any processor complex resources. The cache memory complex includes a first cache memory and a second cache memory that operate in concert to achieve increased RAID parity write efficiency for a RAID Level 3, 4, 5, or 6 write operation. The I/O controller of the present invention also performs substantially concurrent manipulations on the new host data being written to the RAID array and the old host data and old parity data used to generate a new parity data corresponding to the new host data.

Brief Summary Text (2):
This invention relates to the field of Input/Output (I/O) controllers, and in particular to an external I/O bus controller between external I/O busses for an independent access parity disk array system, also known as a Redundant Array of Independent Disks (RAID) system.

Brief Summary Text (4):
Redundant Arrays of Independent Disks (RAID) technology is well known and widely used for its desirable characteristics of cost-effective, high-performance, and high reliability data storage. RAID systems achieve these desirable characteristics by organizing multiple physical disk devices into array configurations that are each viewed by host computer applications as very large capacity virtual disks. For purposes of this discussion, a disk or disk device is a non-volatile, random access, block-addressable, multiple-write multiple-read storage device. Examples of a disk include, but are not limited to, a rotating magnetic disk and/or optical disk, or other non-volatile electronic storage element. Further, a host or host computer is a computer or computer system to which at least one disk is attached and accessible for data storage and I/O. Examples of a host computer include, but are not limited to, a mainframe computer, server, workstation, and personal computer, in addition to multiprocessors and/or computer complexes such as clusters.

Brief Summary Text (5):
One highly desirable property of any storage system is its data reliability, that is, the likelihood of data loss or data survival in the event of a failure in the array. Early RAID configurations were based on striped array and mirrored array configurations. A striped array is one that distributes application data across two or more member disks in the array in a regular pattern. However, without redundancy the striped array configuration is only as reliable as the least reliable member disk in the array. A mirrored array is one where at least two disk members maintain identical images of user data. However, maintaining identical images of user data

is a reliable but extremely costly approach to achieving redundancy. For these reasons, a series of RAID levels 1 through 6 called the Berkley RAID Levels was developed to examine alternative ways to combine disks into arrays having desirable combinations of affordability, data reliability, and I/O performance. Each RAID level was stated in terms of an algorithm for mapping data blocks presented to applications onto an array's physical storage and a mechanism for providing data protection.

Brief Summary Text (6):
The RAID levels that are relevant to the present discussion include, but are not limited to, RAID levels 3 through 6. Characteristics of these RAID levels include, but are not limited to, independent access to user data, and parity. Independent access means that individual member disks in the RAID array can operate independent of each other to the extent that multiple I/O requests to the array can be serviced concurrently, versus parallel access where multiple I/O requests to the array are serviced seriatim by the array as a collective unit.

Brief Summary Text (7):
Parity in a RAID array is a redundancy mechanism to protect user data from loss due to a member disk failure. Types of Parity RAID relevant to the present discussion include distributed parity as in RAID level 5, and independent or single parity as in RAID levels 3, 4, and 6. Independent or single parity means that the blocks of user data are protected by corresponding parity data that is located on a single member disk separate from the user data. Distributed parity means that the blocks of user data are protected by corresponding parity data that are distributed across at least two other member disks separate from the user data.

Brief Summary Text (8):
The redundancy in parity RAID arrays is achieved with a bit-by-bit Exclusive-OR operation on data strips from among the member disks. For example, if a first member disk and a second member disk contain user data in a given data strip, a corresponding strip on a third member disk will contain an Exclusive-OR result of the first and second member disk strips. The Exclusive-OR function is particularly useful for parity RAID arrays because when user data cannot be retrieved due to a failure on a member disk, the location of the unreadable user data is known and can be recovered by computing the Exclusive-OR of the remaining user data from the corresponding strips of the functioning member disks. The computation is easy to perform in software or by an easy-to-implement hardware assist and makes data recovery highly reliable. Thus, the contents of any strip of data on any single one of the member disks in the array can be regenerated from the contents of the corresponding strips on the remaining disks in the array. Similarly, a given strip's contribution to the result of an Exclusive-OR result can be nullified by computing the Exclusive-OR of the result with the strip's contents.

Brief Summary Text (9):
One significant disadvantage that exists in parity RAID implementations, is asymmetrical I/O performance where read operations substantially outperform write operations. In RAID Level 5 for example where parity data is distributed across most or all member disks, a normal data read operation requires simple coordination of tasks that include selecting the member disk or disks that can satisfy the read request, scheduling and executing read requests for as many consecutively addressed strips of data as are needed, and reporting the completion of the overall read request to the host application. However, a normal write operation requires many more tasks that include: 1) reading the target data blocks and holding them in a first temporary buffer; 2) reading the corresponding parity blocks and holding them in a second temporary buffer; 3) generating a restored set of data blocks by removing the target data blocks' contribution to the parity of the corresponding parity blocks as a result of computing the Exclusive-OR of the target data blocks and the parity blocks; 4) generating updated parity blocks from the Exclusive-OR of the application data to be written and the restored set of data blocks from step 3

above; 5) writing the updated parity blocks to the parity disk; and 6) writing the application data to the appropriate target member disk. Further complicating the write operation steps stated above is where the write request maps to more than one member disk because these steps must be performed for each member disk.

Brief Summary Text (10):
To reduce or eliminate the above identified intrinsic read/write asymmetry, parity RAID implementations must be augmented by caching schemes and/or other parallel multiprocessor amendments. One solution to this performance asymmetry is to limit the use of parity RAID implementations to lite-duty read/write uses, read-intensive uses, or extremely large write operations. However, these solutions avoid the problem altogether rather than solving or otherwise improving the I/O performance asymmetry.

Brief Summary Text (11):
Another solution to the I/O performance asymmetry is to implement a non-volatile write-back cache that minimizes the I/O performance impact of momentary I/O overloads. However, this solution only addresses overload situations and not the ongoing read-modify-write cycle demands of normal use.

Brief Summary Text (12):
Another solution to the I/O performance asymmetry is to implement a volatile host data direct store to an I/O bus Central Processing Unit (CPU) memory scheme. In this scheme, there is an I/O bus bridge controller and a master CPU controller between the host system and the RAID array. The bridge controller and the master CPU controller function cooperatively to provide an I/O bus caching system in addition to the necessary processing resources to perform the read-modify-write operations for the parity RAID implementation. Here, the basic idea is that the bridge controller and the master CPU controller both read and/or manipulate the host data from the CPU's memory. Specifically, the read-modify-write operations in this scheme include: 1) the new host data D.sub.n is written to the RAID array is first written to the CPU memory; 2) the data D.sub.n is read from the CPU memory to the bridge controller memory by the bridge controller; 3) the corresponding old disk data D.sub.o is read from the appropriate RAID array disk member into the bridge controller memory and XOR'ed with the D.sub.n data to generate D.sub.o+n data in place of the previous D.sub.n data; 4) the old disk parity data P.sub.o is read from the appropriate RAID array member disk and XOR'ed with the D.sub.o+n data to generate a new parity P.sub.n ; 5) the D.sub.n data is written from the CPU memory to the appropriate RAID array member disk; and 6) the new parity P.sub.n data is written from the bridge controller memory to the appropriate RAID array member disk. However, this scheme is imposes an undesirably heavy burden on the CPU's processing resources by requiring multiple transactions across the CPU bus interface to the CPU memory.

Brief Summary Text (13):
For example only, the overall data throughput performance of a volatile host data direct store scheme can be shown to yield an average of only about 6.9 Mbytes/sec before subtracting performance losses due to overhead which can be significant where CPU bus arbitration holds off the CPU from the CPU memory for multiple bus transactions. This performance estimate is based on {average[(read D.sub.n/bus memory)+(read D.sub.n/member disk)+(read P.sub.n)+(2*rmw D/P.sub.o)]/number of total cycles}=Mbytes/sec. For this reason, the volatile host data direct store to an I/O bus CPU memory scheme is undesirable.

Brief Summary Text (14):
Another solution to the I/O performance asymmetry is to implement a volatile host data direct store to the I/O bus bridge controller's memory. This solution is based on the same hardware configuration and fundamental steps as discussed previously in the volatile host data direct store to a CPU memory scheme, however, additional I/O performance is realized in the bridge controller memory scheme by first writing the

new host data D.sub.n directly to the bridge controller memory and then copying D.sub.n to the CPU memory. This modification to the CPU memory scheme limits CPU memory accesses to only two so that the overall data throughput, for example, is at or about 8.33 Mbytes/sec prior to subtracting overhead. Note that the throughput performance estimates discussed herein are for example purposes only to illustrate a relative performance comparison among different solutions to the I/O performance asymmetry problem.

Brief Summary Text (15):
Although the bridge controller memory scheme offers further improved performance over the CPU memory scheme, there exists a long felt need for increasingly higher performance I/O bus bridge systems that do not adversely impact transparent expandability of an I/O bus system, I/O bus bandwidth, host CPU performance, or overall system performance. Such a solution has heretofore not been known prior to the invention as disclosed and claimed below.

Brief Summary Text (19):
Although the processor complex is the primary means for coordinating input/output operations across the cache memory complex, the processor complex plays no other role in the actual manipulation of host data or parity data toward completing the tasks of writing host data to permanent non-volatile data to a data storage device such as a RAID array using any one of the well known RAID parity data writes of Levels 3-5 or Level 6. The first cache memory and second cache memory of the cache memory complex and their corresponding cache controllers, are the primary means for operationally implementing a data parity write operation across the cache memory complex from the host system to the data storage system independent of the processor complex and/or the processor complex resources.

Brief Summary Text (20):
In a preferred embodiment, the data storage system is a Redundant Array of Independent Disks (RAID) system and the external I/O controller system is an implementation of the RAID Level 5 distributed parity read/write for independent access to a RAID disk array. Further in the preferred embodiment, the cache memory complex includes a new host data cache as the first cache memory and a parity data cache as the second cache memory.

Detailed Description Text (3):
FIG. 1 illustrates a system level block diagram view of a computing system 100 that includes an external I/O controller 101 on I/O bus 118 between a host system 140 and a high-performance peripheral data storage system 150. The host system 140 is operably connected to I/O bus 118 by way of host system interface 147 and host I/O bus 143. In the present example, the high-performance peripheral data storage system 150 includes a common storage system I/O bus 151 that interconnects one or more storage devices 152-154. The storage system 150 is operably connected to I/O bust 118 by way of storage system interface 158 and common storage system I/O bus 151.

Detailed Description Text (4):
FIG. 1 is for example purposes only and is not intended to imply any operational or architectural limitations on the external I/O controller 101 itself or the bus 118, host system 140, or data storage system 150 to which it is attached. The preferred embodiment of the external I/O bus controller 101 joins two I/O busses such as the Peripheral Component Interconnect (PCI) bus architecture commonly used in Personal Computers (PC's) in combination with Small Computer System Interface (SCSI) host systems and/or high-performance peripheral data storage systems. For purposes of the present disclosure, high-performance peripheral data storage systems include, but are not limited to, Redundant Arrays of Independent Disks (RAID) configurations. RAID refers to any of the family of methods and apparatus used for purposes including, but not limited to, managing multiple independent disks or other random access memory or partitions thereof in a manner that achieves a

desired level of availability, efficiency, capacity, performance, and economic cost of storing large quantities of readily available data in a data storage system.

Detailed Description Text (8):
In a preferred embodiment, the cache memory complex is used to implement a RAID Level 3,4, 5, or 6 parity write implementation. More particularly, the cache memory complex configuration is ideally used for a RAID Level 5 write operation for an independent access disk array having distributed parity. The RAID Level 5 supports the more robust Exclusive-OR (XOR) type parity that can be implemented as a more reliable alternative to existing memory shadowing techniques. In summary, the contents of any one block of data in a partition of the total data cache memory area of a RAID Level 5 implementation can be regenerated from the contents of corresponding blocks in the remaining memory area by performing an XOR of the individual strips of data around the target block. The basic XOR parity concept is well known in the data redundancy art, however, specific implementations can vary widely in terms of performance.

Detailed Description Text (11):
FIG. 2 illustrates a timing sequence 200 for a write operation to an independent access distributed parity disk array 150 by way of the I/O controller system 101. The timing illustration is presented to emphasize the simultaneous and/or substantially concurrent tasks that can occur between the first cache memory 120 and the second cache memory 130 during a RAID parity write operation. Key to appreciating the efficiencies that can be realized by the present invention is that three data components must be manipulated to successfully complete the write operation in a manner that is transparent to the user. The three data components include the new host data D.sub.n, the old parity data P.sub.o, and the old host data D.sub.o.

Detailed Description Text (19):
At step 350, the resulting new parity data P.sub.n is written from the second cache memory 130 to permanent non-volatile memory in data storage system 150. The new parity data P.sub.n can be used for generating subsequent new parity data for future write operations of the corresponding host data strips. The new parity data P.sub.n may also be used for data recovery purposes in a manner that is widely known in the RAID parity data recovery of Levels 3-6. Upon completion of steps 335-350, processing for the present parity write operation is compete at step 352.

Detailed Description Text (20):
The following non-limiting example is presented to facilitate a concrete performance comparison between existing I/O controller systems and the I/O controller system of the present invention. In a computing environment such as with an I/O bus 118 between the host system 140 and the data storage system 150 for example, where the I/O bus 118 operates at or about 133 Mbytes/second and where the cache memory accesses operate at our about 80 Mbytes/second, one performance estimate for a RAID Level 5 parity write using the present invention can be at or about 11.1 Mbytes/second less overhead. This estimate is for comparative example purposes only and is based on a 40 Mbyte/second Exclusive-OR operation performed "on-the-fly" {average[(read D.sub.n/host)+(read D.sub.n/cache2)+(read P.sub.n)+(2*rmw D/P.sub.o)]/number of total cycles}=Mbytes/seconds-overhead. That is, for a 6 cycle write operation, the average throughput is {average [(80)+(80)+(80)+(80)+(2*40)]/6}=11.1 Mbytes/second. Thus, the I/O controller of the present invention affords a significantly enhanced performance over existing I/O controllers.

Detailed Description Text (22):
The external I/O controller system includes a first cache memory and a second cache memory that are operationally independent from the processor complex and can asynchronously execute the individual tasks necessary to complete a RAID parity write to a RAID array independent of the processor complex components and/or processor complex resources, in a manner that is faster and more efficient than if

processor complex resources were used. Although specific embodiments are disclosed herein, it is expected that persons skilled in the art can and will make, use, and/or sell alternative external I/O controller systems that are within the scope of the following claims either literally or under the Doctrine of Equivalents.

CLAIMS:

6. The system of claim 1, wherein said means for operationally implementing is a Redundant Array of Independent Disk parity data implementation selected from at least one of a group comprised of: a Level 3 RAID, a Level 4 RAID, a Level 5 RAID, and a Level 6 RAID.

12. A Redundant Array of Independent Disks (RAID) controller system on an input/output bus between a host system and at least one data storage device in a RAID array, said RAID controller system having a processor complex and a cache memory complex, said processor complex having processor complex resources that include a central processing unit and a central processing unit memory, said RAID controller system comprising:

said cache memory complex being operationally independent of said processor complex and having a first cache controlled by a first cache controller and a second cache controlled by a second cache controller, wherein said second cache is independently controlled from said first cache and said first cache controller;

means for executing a RAID parity write operation across said cache memory complex from said host system to said RAID array absent any host data manipulation and parity data manipulation with said processor complex resources, said means for executing said RAID parity write operation including:

means for writing new host data to said first cache and said second cache;

means for writing said new host data from said first cache to said at least one data storage device by said first cache controller;

means for generating new parity data in said second cache for said new host data at a time concurrent with the writing of said new host data from said first cache to said at least one data storage device; and

means for writing said new parity data from said second cache to said at least one data storage device by said second cache controller.

L16: Entry 1 of 2 File: USPT Apr 29, 2003

DOCUMENT-IDENTIFIER: US 6557039 B1
TITLE: System and method for managing information retrievals from distributed archives

Parent Case Text (2):
This application is based on and claims priority to U.S. Provisional Patent Application No. 60/108,245, filed Nov. 13, 1998, entitled SYSTEM FOR MANAGING INFORMATION RETRIEVALS FROM DISTRIBUTED DOCUMENT ARCHIVES ON A GLOBAL BASIS, the entire disclosure of which is hereby incorporated by reference.

Brief Summary Text (5):
The hardware typically incorporated in an electronic archive is comprised of a general purpose computer and storage devices (such as magnetic disks, optical disks and magnetic tape subsystems). The hardware is typically operated and accessed by software comprising an operating system, database management systems, hierarchical storage management software (HSM) and archive management software. There are at least four significant limitations associated with current long term archival systems. First, larger corporations will invariably require several geographically diverse heterogenous archival systems in order to support the various operations of the corporation throughout the country and the world. For example, The corporation's research and development facility in London England has a separate archival system from the archival system for one of the corporation's manufacturing sites in Dallas Tex. Even if each of the archive facilities has a heterogeneous archival (e.g., a database manager) the hardware and the software comprising the archival at the two sites is invariably provided by two different vendors whose proprietary product are not interoperable (i.e., the software at the London site cannot be used to access the information stored at the Dallas site).

Detailed Description Text (3):
Schematically included in each of the archives 100-106 are the physical storage devices 110, the software 112 for accessing the physical devices 110, the site specific software 114 for controlling access to archived information, and site specific messaging system 116 for communication with a site. Typical storage devices 102 include Direct Access Storage Devices (DASD), optical storage devices and magnetic tape devices. These storage devices are typically configured in a hierarchical manner such that information that is more recent or that is more often accessed is stored on devices with the quickest access time, for example DASD. Using conventional archiving techniques, as electronic information "ages", it is migrated for archival purposes from DASD to devices with a slower access time such as optical disks or magnetic tape. Optical disks and tape provide a cost effective means for the storage of large quantities of electronic information. Tapes are typically stored and accessed through tape silos while a large quantity of optical disks are stored and accessed from one or more jukeboxes. Some specific examples of storage devices 110 include IBM and EMC magnetic disks, STK magnetic tape silos, Boxhill RAID magnetic disks, and Hewlett Packard magneto-optical jukeboxes.

Detailed Description Text (10):
FIG. 3 depicts an overview of the processing and flow of information through archive access system 140. In step 300 the BAIM Input Processing section 205 (see FIG. 2) receives and processes information requests from users. It is determined in

step 310 what type of information is being requested. If the information is data which can be located using database indexes, the request is forwarded to the AIM module 235 in step 320 which retrieves the data from the archives 100-106. Upon retrieval of the data, it is processed in step 330 for presentation to the user by the Output Processing module 210 of the BAIM 200. In a preferred embodiment, this retrieval occurs in two steps. First, the relevant index is retrieved and presented to the user (e.g., the user requests to see checks for the month of August from a particular account). When the user selects particular data items to view from the retrieved index, the system 140 retrieves the actual data for presentation to the user. Since indexed data is typically stored on DASD (quick retrieval time) the more complex retrieval process (e.g., prioritization) employed for the retrieval of documents described below is not required but could be used.

Detailed Description Text (19):
As the Core Processing Block 220 receives requests for information retrievals from the BAIM Input Processing section 205, the requests are queued by the Queue Management module 220. The service class contained in the request from the user 150, 160 is used by the Queue Management module 220 to set the priority the request. If the Queue Management module 220 has calculated that there will be a delay with respect to fulfilling the request (with respect to the priority indicated by the user in the assignment of the service class) the Queue Management module 220 sends an advice message to the output queue (see below) for immediate delivery to the requesting application.

Detailed Description Text (20):
The Process Retrieval Fulfillment module 255 is responsible for processing the requests from the queues established by the Queue Management module 220. The process followed by the Retrieval Fulfillment module 225 is illustrated in FIG. 5. In step 500, the Retrieval Fulfillment module 225 retrieves the request with the highest priority from the queue. In step 510, it is determined whether the requested information has previously been retrieved and is already cached by the system 140. The caching feature of the present invention is more fully described below with respect to the cache control module 255 (see FIG. 2). If the information is not found in the cache, the Retrieval Fulfillment module 225 calls the Archive Interface manager 235 to perform the actual retrieval function (see FIG. 6 and associated description). If the information has been cached, the Retrieval Fulfillment module 225 retrieves the document or file from the cache in step 530 and returns it with a message to the Output Control 230 of the CPB 215 for eventual transmittal back to the requesting user 150, 160 as described below. As with any substantive action by system 140, the audit log is updated in step 540.

Detailed Description Text (29):
The Priority Administration section 250 allows the manual intervention to change the priority number (01-99) for an individual request or a group of requests. This function allows dynamic priority re-assignment during periods where heavy request volumes are creating request backlogs.

CLAIMS:

9. The system according to claim 7, wherein the administrative manager further comprises a priority administration section, the priority administration section dynamically re-assigning priorities of the requests for information.

39. The method according to claim 27, further comprising the step of assigning a priority to the request for information.

40. The method according to claim 39, wherein a plurality of requests for information are received and assigned priorities, the method further comprising the step of dynamically re-assigning priorities of the requests for information.

h e b b g e e e f c e h e ge

5146605	September 1992	Beukema et al.	395/575
5155845	October 1992	Beal et al.	395/575
5285451	February 1994	Henson et al.	371/11.1
5491787	February 1996	Hashemi	714/11
5633999	May 1997	Clowes et al.	395/182.04
5758052	May 1998	Glowny et al.	714/12
5812748	September 1998	Ohran et al.	395/182.02
5926619	July 1999	Badovinatz et al.	714/4
5951695	September 1999	Kolovson	714/11
5987621	November 1999	Duso et al.	714/4
6065053	May 2000	Nouri et al.	709/222
6108300	August 2000	Coile et al.	340/825.01
6145101	November 2000	Pike	714/46
6230200	May 2001	Forecast et al.	709/219
6292905	September 2001	Wallach et al.	709/239
6327670	December 2001	Hellenthal et al.	714/10
6363464	March 2002	Mangione	710/53

FOREIGN PATENT DOCUMENTS

FOREIGN-PAT-NO	PUBN-DATE	COUNTRY	US-CL
WO 97/16023	May 1997	WO	

OTHER PUBLICATIONS

D.L. Burkes and R.K. Treiber, "Design Approaches for Real-Time Transaction Processing Remote Site Recovery", Digest of Papers: Thirty-Fifth IEEE Computer Society International Conference, Feb. 26-Mar. 2, 1990, pp. 568-572.

ART-UNIT: 2184

PRIMARY-EXAMINER: Baderman; Scott

ASSISTANT-EXAMINER: Lohn; Joshua

ATTY-AGENT-FIRM: Howry Simon Arnold & White LLP

ABSTRACT:

A file server includes a plurality of stream server computers linking data storage to a data network, and at least two controller servers for controlling the stream server computers. The controller servers are programmed so that at a given time one of the controller servers is active in controlling the stream server computers, and another of the controller servers is inactive. The inactive controller server is programmed to respond automatically to a failure of the active controller server by becoming active. For example, each of the controller servers has a respective flag for indicating whether or not the controller server is active. Each controller server is programmed so that, upon booting, it will read the flag of the other stream server, and if the flag of the other controller server indicates that the other controller server is active, then the controller server becomes inactive. Otherwise, the stream server assumes an active or inactive state based on a predetermined arbitration method. The active controller server also reports failure

conditions to a customer service center, monitors the inactive controller server to ensure that the inactive controller server is in a state of readiness, and monitors itself to determine whether it should become inactive.

24 Claims, 45 Drawing figures

☐ 2. Document ID: US 6330572 B1

L11: Entry 2 of 16 File: USPT Dec 11, 2001

US-PAT-NO: 6330572
DOCUMENT-IDENTIFIER: US 6330572 B1

TITLE: Hierarchical data storage management

DATE-ISSUED: December 11, 2001

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Sitka; Larry	Stillwater	MN		

ASSIGNEE-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY	TYPE CODE
Imation Corp.	Oakdale	MN			02

APPL-NO: 09/ 354280 [PALM]
DATE FILED: July 15, 1999

PARENT-CASE:
This application claims priority from U.S. Provisional Application Serial No. 60/092,853, filed Jul. 15, 1998, the entire content of which is incorporated herein by reference.

INT-CL: [07] G06 F 17/30

US-CL-ISSUED: 707/205; 203/204, 203/104, 203/100
US-CL-CURRENT: 707/205; 707/100, 707/104.1, 707/203, 707/204

FIELD-OF-SEARCH: 707/104, 707/100, 707/201, 707/205, 707/203-204

PRIOR-ART-DISCLOSED:

U.S. PATENT DOCUMENTS

PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL
5276860	January 1994	Fortier et al.	
5276867	January 1994	Kenley et al.	
5317728	May 1994	Tevis et al.	
5367698	November 1994	Webber et al.	

h e b b g e e e f e ef b e

5537585	July 1996	Blickenstaff et al.	707/100
5557790	September 1996	Bingham et al.	
5579507	November 1996	Hosouchi et al.	
5579516	November 1996	Van Maren et al.	395/601
5584008	December 1996	Shimada et al.	
5778395	July 1998	Whiting et al.	707/204
5784646	July 1998	Sawada	
5822780	October 1998	Schutzman	
5832522	November 1998	Blickenstaff et al.	
5878410	March 1999	Zbikowski et al.	707/2
5918229	June 1999	Davis et al.	707/10
5991753	November 1999	Wilde	
6026474	February 2000	Carter et al.	711/202
6108713	January 2000	Coli et al.	705/2

FOREIGN PATENT DOCUMENTS

FOREIGN-PAT-NO	PUBN-DATE	COUNTRY	US-CL
0 474 395 A2	November 1992	EP	

OTHER PUBLICATIONS

"Hierarchical View of Filesets," IBM Technical Disclosure Bulletin, vol. 36(5):167 (1993).
Shiers, J.D., "Data Management at Cern: Current Status and Future Trends," http://www.computer.org/conferences/mss95/shiers/shiers.htm, Cern, Geneva, Switzerland, pp. 1-11 (1995).
"Graphical User Interface for the Distributed Computing Environment," IBM Technical Disclosure Bulletin, vol. 38(1):409-410 (1995).

ART-UNIT: 212

PRIMARY-EXAMINER: Alam; Hosain T.

ASSISTANT-EXAMINER: Truong; Cam-Y

ATTY-AGENT-FIRM: Bauer; William D.

ABSTRACT:

A system and method for managing the storage of files within an HSM system incorporate an architecture and methodology that facilitate the storage and retrieval of large image files as part of an overall image processing workflow. In particular, the system and method may find ready application in a workflow that involves the processing of groups of images associated with particular customers, projects, or transactions, and may act as a storage server for a client application that implements the workflow. The system and method may be useful, for example, in handling the storage of images uploaded from scanned photographic film, or digital images submitted to a photo-processing shop by amateur or professional photographers. In this case, the client application can be a photo-processing application that could provide for various media formats, sizes, and quantities of image reproductions for a consumer. As another example, the system and method may be useful in handling the storage of medical diagnostic images associated with a

particular medical patient or study. In this case, the client application could be a picture archival communication system (PACS) that manages the archival of imagery for viewing by physicians. Further, the system and method may be useful in handling the storage of images associated with particular printing jobs, e.g., for publishers, advertising customers, and the like. In this case, the client application could be a digital prepress workflow application.

32 Claims, 8 Drawing figures

☐ 3. Document ID: US 6230200 B1

L11: Entry 3 of 16 File: USPT May 8, 2001

US-PAT-NO: 6230200
DOCUMENT-IDENTIFIER: US 6230200 B1

TITLE: Dynamic modeling for resource allocation in a file server

DATE-ISSUED: May 8, 2001

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Forecast; John	Newton	MA		
Duso; Wayne W.	Shrewsbury	MA		

ASSIGNEE-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY	TYPE CODE
EMC Corporation	Hopkinton	MA			02

APPL-NO: 08/ 925026 [PALM]
DATE FILED: September 8, 1997

INT-CL: [07] G06 F 15/16

US-CL-ISSUED: 709/226; 709/219
US-CL-CURRENT: 709/226; 709/219

FIELD-OF-SEARCH: 395/200.49, 395/200.56, 395/200.55, 395/200.61, 395/311, 395/154, 395/500, 709/225, 709/219, 709/238, 709/216, 707/3, 711/114, 711/118, 364/578, 364/514, 348/390, 348/419, 706/14

PRIOR-ART-DISCLOSED:

U.S. PATENT DOCUMENTS

PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL
4591983	May 1986	Bennett et al.	364/403
4658370	April 1987	Erman et al.	364/513
4890227	December 1989	Wanatabe et al.	706/14

h e b b g e e e f e ef b e

5045852	September 1991	Michell et al.	341/51
5461611	October 1995	Drake, Jr. et al.	370/420
5526414	June 1996	Bedard et al.	379/221
5544313	August 1996	Shachnai et al.	395/200.49
5544327	August 1996	Dan et al.	395/200.64
5568602	October 1996	Callahan et al.	395/154
5583995	December 1996	Gardner et al.	395/200.49
5630067	May 1997	Kindell et al.	395/200.61
5638516	June 1997	Duzett et al.	709/238
5646676	July 1997	Dewkett et al.	348/7
5694170	December 1997	Tiwari et al.	348/390
5719632	February 1998	Hoang et al.	348/419
5737747	April 1998	Vishlitzky et al.	711/118
5764961	June 1998	Bhat	395/500
5796966	August 1998	Simcoe et al.	395/311
5802301	September 1998	Dan et al.	395/200.53
5832222	November 1998	Dziadosz et al.	709/216
5850352	December 1998	Moezzi et al.	364/514
5892915	April 1999	Duso et al.	709/219
5907837	May 1999	Ferrel et al.	707/3
5917730	June 1999	Rittie et al.	364/578
5933603	August 1999	Vahalia et al.	709/225
5974503	October 1999	Venkatesh et al.	711/114
6006018	December 1999	Brunett et al.	709/219
6061504	May 2000	Tzelnic et al.	709/219

FOREIGN PATENT DOCUMENTS

FOREIGN-PAT-NO	PUBN-DATE	COUNTRY	US-CL
WO 97/16023	May 1997	WO	

OTHER PUBLICATIONS

Pan et al. A Time-Scale Dependent Disk Scheduling Scheme for Multimedia-on-Demand Servers. IEEE Sep. 1996.*
Jadav et al. Design & Evaluation of Data Access Strategies in a High Performance Multimedia-on-Demand Server. IEEE 1995.*
Christodoulakis et al. "Data organization and storage hierarchies in a multimedia server," IEEE, 1993.*
Vin et al., "An observation-based admission control algorithm for multimedia servers," IEEE, 1994.*
Ghafir et al., "Multimedia servers-design and performance," IEEE, 1994.*
Gemmel et al., "Multimedia storage servers: a tutorial," IEEE, 1995.*
Nishikawa et al., "High-performance VOD server AIMS," IEEE, 1995.*
Molano et al., "The design and implementation of a multimdedia storage server to support video-on-demand applications," IEEE, 1996.*
Jadav et al., "Techniques for increasing the stream capacity of a multimedia server," IEEE, 1996.*
Kyeongho et al., "Scheduling of storage and cache servers for replicated multimedia data," IEEE, 1997.*

h e b b g e e e f e ef b e

Wu et al., "Scheduling for interactive operations in parallel video servers," IEEE, 1997.*
Shau et al, "On the efficient retrieval of VBR video in a multimedia server," IEEE, 1997.*
Disz et al., "Performance model of Argonne Voyager multimedia server," IEEE, 1997.*

Serpanos et al., "MMPacking: A load and storage balancing algorithm for distributed multimedia servers," IEEE, 1997.*
TINA, "A Common Software Architecture for Multimedia and Information Services", Emmanuel Darmois, Motoo Hoshi, http://www.tinac.com/about/nutshell.htm (Aug. 9, 1997).

ART-UNIT: 212

PRIMARY-EXAMINER: Harrell; Robert B.

ASSISTANT-EXAMINER: Vu; Thong

ATTY-AGENT-FIRM: Howrey Simon Arnold & White, LLP

ABSTRACT:

Resources in a file server are allocated by dynamically modeling a configuration of data handling components in the file server and routings of data streams through the data handling components. The dynamic model is a computer model maintained in memory by a controller of the file server. For example, the dynamic model is a directed acyclic graph in which nodes represent the data handling components and edges represent data stream paths. Each node has a list of resources and current allocations of the resources. Associated with each active data stream is a list of pointers to the nodes and current allocations for the data stream. The controller of the file server has programs for automatically creating the dynamic model, modifying the dynamic model in response to component changes such as component failures, enforcing a scheduling and admissions policy by allocating resources for a path for a data stream during a search through the dynamic model in response to a client request for data access, de-allocating resources in response to an end-of-stream condition, and balancing allocations of resources to data streams in order to free resources to allocate a path for a requested data stream. The dynamic model is created automatically by collecting information about what components are installed in the file server, the resources of the installed components, and connections between the installed components.

46 Claims, 50 Drawing figures

☐ 4. Document ID: US 6061504 A

L11: Entry 4 of 16 File: USPT May 9, 2000

US-PAT-NO: 6061504
DOCUMENT-IDENTIFIER: US 6061504 A

TITLE: Video file server using an integrated cached disk array and stream server computers

DATE-ISSUED: May 9, 2000

h e b b g e e e f e ef b e

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Tzelnic; Percy	Concord	MA		
Vahalia; Uresh K	Newton	MA		
Ting; Dennis P. J.	Groton	MA		
Vaitzblit; Lev	Concord	MA		
Duso; Wayne W	Shrewsbury	MA		
Alagappan; Kannan M	Arlington	MA		
Forecast; John	Newton	MA		

ASSIGNEE-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY	TYPE CODE
EMC Corporation	Hopkinton	MA			02

APPL-NO: 08/ 661152 [PALM]
DATE FILED: June 10, 1996

PARENT-CASE:
RELATED APPLICATIONS The present application is a divisional of provisional application Ser. No. 60/005,988 filed Oct. 27, 1995 by Percy Tzelnic et al., entitled "Video File Server," fully incorporated herein by reference. The detailed description in the present application has been updated to conform to the applicants' present implementation of their invention. The present application also contains a disclosure which is common with the following other divisional applications of Ser. No. 60/005,988: Natan Vishlitzky et al., Ser. No. 08/661,185 filed Jun. 10, 1996, entitled "Prefetching to Service Multiple Video Streams from an Integrated Cached Disk Array", issued on Apr. 17, 1998 as U.S. Pat. No. 5,737,747; Uresh Vahalia et al., Ser. No. 08/661,053 filed Jun. 10, 1996 entitled "Staggered Stream Support for Video On Demand" now U.S. Pat. No. 5,933,603; and Percy Tzelnic et al., Ser. No. 08/661,187 filed Jun. 10, 1996, entitled "On-Line Tape Backup Using an Integrated Cached Disk Array," and now U.S. Pat. No. 5,829,046.

INT-CL: [07] G06 F 15/16

US-CL-ISSUED: 395/200.49; 395/200.33, 395/872, 711/113
US-CL-CURRENT: 709/219; 709/203, 710/52, 711/113

FIELD-OF-SEARCH: 395/200.49, 395/200.48, 395/200.47, 395/200.31, 395/200.32, 395/200.33, 395/182.04, 395/600, 395/200.64, 395/872-877, 395/894, 364/514A, 711/113

PRIOR-ART-DISCLOSED:

U.S. PATENT DOCUMENTS

PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL
4608688	August 1986	Hansen et al.	371/11
4755928	July 1988	Johnson et al.	364/200
5175837	December 1992	Arnold et al.	395/425
5206939	April 1993	Yanai et al.	395/400
5208665	May 1993	McCalley et al.	358/862

h e b b g e e e f e ef b e

5218695	June 1993	Noveck et al.	395/600
5255270	October 1993	Yanai et al.	371/10.2
5269011	December 1993	Yanai et al.	395/425
5276860	January 1994	Fortier et al.	395/575
5276867	January 1994	Kenley et al.	395/600
5335352	August 1994	Yanai et al.	395/800
5341493	August 1994	Yanai et al.	395/425
5367698	November 1994	Webber et al.	395/800
5371532	December 1994	Gelman et al.	348/7
5381539	January 1995	Yanai et al.	395/425
5408465	April 1995	Gusella et al.	370/17
5410343	April 1995	Coddington et al.	348/7
5414455	May 1995	Hooper et al.	348/7
5442390	August 1995	Hooper et al.	348/7
5442749	August 1995	Northcutt et al.	395/200.09
5442771	August 1995	Filepp et al.	395/200.08
5477263	December 1995	O'Callaghan et al.	348/7
5504873	April 1996	Martin et al.	395/438
5508732	April 1996	Bottomley et al.	348/7
5508733	April 1996	Kassatly	348/13
5519435	May 1996	Anderson	348/8
5528282	June 1996	Voeten et al.	348/7
5528513	June 1996	Vaitzblit et al.	364/514A
5530557	June 1996	Asit et al.	358/342
5533021	July 1996	Branstad et al.	370/60.1
5534912	July 1996	Kostreski	348/6
5537408	July 1996	Branstad et al.	370/79
5539660	July 1996	Blair et al.	364/514C
5544313	August 1996	Shachnai et al.	395/200.01
5544327	August 1996	Dan et al.	395/250
5544345	August 1996	Carpenter et al.	395/477
5544347	August 1996	Yanai et al.	395/489
5550577	August 1996	Verbiest et al.	348/7
5550982	August 1996	Long et al.	395/200.13
5551025	August 1996	O'Reilly et al.	395/600
5553005	September 1996	Voeten et al.	364/514R
5557317	September 1996	Nishio et al.	348/7
5559949	September 1996	Reimer et al.	395/161
5574662	November 1996	Windrem et al.	364/514R
5583561	December 1996	Baker et al.	348/7
5586264	December 1996	Belknap et al.	395/200.08
5590320	December 1996	Maxey	395/619
5594910	January 1997	Filepp et al.	395/800
5603058	February 1997	Belknap et al.	395/855
5606359	February 1997	Youden et al.	348/7
5610653	March 1997	Abecassis	348/110
5625405	April 1997	DuLac et al.	348/7

5633810	May 1997	Mandal et al.	364/514
5633999	May 1997	Clowes et al.	395/182.04
5649103	July 1997	Datta et al.	395/200.33
5720037	February 1998	Biliris et al.	345/327
5734828	March 1998	Pendse et al.	395/200.33
5737495	April 1998	Adams et al.	707/104
5737747	April 1998	Vishlitzky et al.	711/118
5829046	October 1998	Tzelnic et al.	711/600

FOREIGN PATENT DOCUMENTS

FOREIGN-PAT-NO	PUBN-DATE	COUNTRY	US-CL
0 061 570 A3	October 1982	EP	
633694	January 1995	EP	
0 683 464 A2	November 1995	EP	
0 697 660 A1	February 1996	EP	
0 701 372 A1	March 1996	EP	
WO93/16557	August 1993	WO	
WO 94/00816	January 1994	WO	
WO95/10918	April 1995	WO	

OTHER PUBLICATIONS

M(aurice) William Collins, "A Network File Storage System," IEEE Seventh Symposium on Mass Storage Systems, Nov. 4-7, 1985, Tuscon, AZ, p. 1-11, Los Alamos Nat. Lab. No. LA-UR-85-3183.
John H. Howard, "An Overview of the Andrew File System," USENIX Winter Conference, Feb. 9-12, 1988, Dallas, TX, p. 23-26.
John H. Howard et al., "Scale and Performance in a Distributed File System," ACM Transactions on Computer Systems, vol. 6, No. 1, Feb. 1988, p. 51-81.
David C. Steere et al., "Efficient User-Level File Cache Management on the Sun Vnode Interface," USENIX Summer Conference, Jun. 11-15, 1990, Anaheim, CA, p. 325-331.
Matt Blaze et al., "Long-Term Caching Strategies for Very Large Distributed File Systems," USENIX, Summer '91, Nashville, TN, p. 3-15.
Thomas W. Page, Jr., et al., "Management of Replicated Volume Location Data in the Ficus Replicated File System," USENIX, Summer '91, Nashville, TN, p. 17-29.
Storage Computer Corporation, "High Performance, Fault Tolerant Disk Array Platform For File Servers And Computer Systems," 1991, Nashua, NH, 12 pages.
France Telecom, "Telesauvegarde," 26 page paper dated Nov. 2, 1994 about work based on France Telecom patent application "Dispositif et Proceed de Sauvegaude a Distance de Donnees Numeriques" [Method and Apparatus for Safeguarding at a Distance Numeric Data], Institute National de la Propiete Industrielle, France, Appln. No. 93.12771, Oct. 28, 1994 (filed Oct. 21, 1993), and partial English translation (15 pages).
Krishnan Natarajan, "Video Servers Take Root," IEEE Spectrum, Apr. 1995, IEEE New York, NY, p. 66-69.
K. K. Ramakrishnan et al., "Operating system support for a video-on-demand file service," Multimedia Systems (1995) 3:53-65.
Ralf Steinmetz, "Analyzing the Multimedia Operating System," IEEE MultiMedia, Spring 1995, p. 68-84.
Audrey Chou, "EMC, Computer-Storage Leader, Still Hears Footsteps," The Wall Street Journal, Aug. 9, 1995, Dow Jones & Co., Princeton, N.J.

Pardhu Vadlamudi, "EMC Hunts for Solutions to Bottlenecks," InfoWorld, Apr. 15, 1996, #1590, San Mateo, CA 94402.
Michael Goldberg, "EMC to Pump Data Over Networks," Computerworld, Apr. 15, 1996.
"EMC Moves Into Servers," Broadcasting Cable, Apr. 15,1996.
"Symmetrix Model 55XX Product Manual, P/N 200-810-550 Rev D," EMC Corporation, Hopkinton, Mass., May 1994, pp. 1-236.
"NFS: Network File System Protocol Specification," RFC 1094, Sun Microsystems, Inc., Mar. 1989, pp. 1-27.
J. Case, M. Fedor, M. Schoffstall, J Davin, "A Simple Network Management Protocol (SNMP)," May 1990, MIT Laboratory for Computer Science, Cambridge, Mass., pp. 1-35.

Rangen PV, Vin HM, "Designing file systems for digital audio and video," Proceedings of the 13th ACM Symposium on Operating Systems Principles, Monterey, Calif., 1991, pp. 81-94.
Vin HM, Rangan PV, (1993), "Designing a multiuser HDTV storage service," IEEE Journal on Selected Areas in Communication, vol. 11, No. 1, Jan. 1993, pp. 153-164.

Anderson DP, Osawa Y, Govindan R, "A file system for continuous media," ACM Transactions on Computer Systems, vol. 10., No. 4, Nov. 1992, pp. 311-337.
Federighi C, "A Distributed Hierarchical Storage Manager for a Video-on-Demand System," Department of Electrical Engr. and Computer Science, University of California, Berkeley, California, Dec. 1993.
Hastings R, The stark continuous-media file server. Proceedings IEEE COMPCON 93, San Francisco, Calif., 1993, pp. 12-15.
Little TD, Rhanger G, Folz RJ, Gibbon JF, Reeve FW, Schelleng DH, Venkatesh D, "A digital on-demand video service supporting content based queries," Proceedings of ACM Multimedia 93, Anaheim, Calif., Aug. 1-6, 1993, pp. 427-436.
Lougher, P, Sheperd, D. "The design of a storage server for continuous media," The Computer Journal, vol. 36, No. 1, 1993, pp. 32-42.
Rangan PV, Vin HM, Ramanathan S, "Designing an on-demand multimedia service," IEEE Communications Magazine, vol. 30, No. 7, Jul. 1992, pp. 56-64.
Sincoskie, WD, "System architecture for a large scale video on demand service," Computer Networks and ISDN Systems, vol. 22, No. 2, Sep. 1991, pp. 155-162.
Tobagi FA, Pang J, "StarWorks (Trademark)--A video applications server," Proceedings, IEEE COMPCON 93, San Francisco, Calif., 1993, pp. 4-11.
Vaitzblit L, "The design and implementation of a high band-width file service for continuous media," Master's Thesis, Massachusetts Institute of Technology, Cambridge, Mass., Nov. 4, 1991.

ART-UNIT: 271

PRIMARY-EXAMINER: Black; Thomas G.

ASSISTANT-EXAMINER: Le; Uyen

ATTY-AGENT-FIRM: Arnold White & Durkee

ABSTRACT:

A video file server includes an integrated cached disk array storage subsystem and a plurality of stream server computers linking the cached disk storage system to the data network for the transfer of video data streams. The video file server further includes a controller server for applying an admission control policy to client requests and assigning stream servers to service the client requests. The stream servers include a real-time scheduler for scheduling isochronous tasks, and supports at least one industry standard network file access protocol and one file access protocol for continuous media file access. The cached disk storage subsystem is responsive to video prefetch commands, and the data specified for a prefetch command for a process are retained in an allocated portion of the cache memory from

the time that the cached disk storage subsystem has responded to the prefetch command to the time that the cached disk storage subsystem responds to a fetch command specifying the data for the process. The time between prefetching and fetching is selected based on available disk and cache resources. The video file server provides video-on-demand service by maintaining and dynamically allocating sliding

windows of video data in the random access memories of the stream server computers.

40 Claims, 20 Drawing figures

☐ 5. Document ID: US 5987621 A

L11: Entry 5 of 16 File: USPT Nov 16, 1999

US-PAT-NO: 5987621
DOCUMENT-IDENTIFIER: US 5987621 A

TITLE: Hardware and software failover services for a file server

DATE-ISSUED: November 16, 1999

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Duso; Wayne W.	Shrewsbury	MA		
Forecast; John	Newton	MA		
Gupta; Uday	Westford	MA		
Vahalia; Uresh K	Newton	MA		
Ting; Dennis P. J.	Groton	MA		

ASSIGNEE-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY	TYPE CODE
EMC Corporation	Hopkinton	MA			02

APPL-NO: 08/ 851507 [PALM]
DATE FILED: May 5, 1997

PARENT-CASE:
RELATED APPLICATIONS The present application is a continuation of provisional application Ser. No. 60/044,948 filed Apr. 25, 1997 by Dinesh Venkatesh, Wayne W. Duso, John Forecast, Uday Gupta, Uresh K. Vahalia, and Dennis P. J. Ting, entitled "Raid Striping, Client-Server Protocols, and Failover Services for a Video File Server."

INT-CL: [06] G06 F 11/00

US-CL-ISSUED: 714/4; 395/200.54
US-CL-CURRENT: 714/4; 709/224

FIELD-OF-SEARCH: 395/182.02, 395/181, 395/182.04, 395/182.05, 395/182.06, 395/182.13, 395/182.18, 395/183.07, 395/183.19, 395/728, 395/734, 395/600, 395/650,

h e b b g e e e f e ef b e

371/40.15, 714/2, 714/3, 714/4, 714/6, 714/8, 714/11, 714/13, 714/21, 714/25, 714/41, 714/47, 714/770

PRIOR-ART-DISCLOSED:

U.S. PATENT DOCUMENTS

PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL
4608688	August 1986	Hansen et al.	371/11
4755928	July 1988	Johnson et al.	364/200
4888686	December 1989	Sinz et al.	364/200
5051887	September 1991	Berger et al.	364/200
5146605	September 1992	Beukema et al.	395/575
5155845	October 1992	Beal et al.	395/575
5175837	December 1992	Arnold et al.	395/425
5206939	April 1993	Yanai et al.	395/400
5208665	May 1993	McCalley et al.	358/862
5218695	June 1993	Noveck et al.	395/600
5255270	October 1993	Yanai et al.	371/10.2
5269011	December 1993	Yanai et al.	395/425
5276860	January 1994	Fortier et al.	395/575
5276867	January 1994	Kenley et al.	395/600
5285451	February 1994	Henson et al.	371/11.1
5335352	August 1994	Yanai et al.	395/800
5341493	August 1994	Yanai et al.	395/425
5367698	November 1994	Webber et al.	395/800
5371532	December 1994	Gelman et al.	348/7
5381539	January 1995	Yanai et al.	395/425
5408465	April 1995	Gusella et al.	370/17
5410343	April 1995	Coddington et al.	348/7
5414455	May 1995	Hooper et al.	348/7
5442390	August 1995	Hooper et al.	348/7
5442749	August 1995	Northcutt et al.	395/200.09
5442771	August 1995	Filepp et al.	395/200.08
5477263	December 1995	O'Callaghan et al.	348/7
5508733	April 1996	Kassatly	348/13
5519435	May 1996	Anderson	348/8
5528282	June 1996	Voeten et al.	348/7
5528513	June 1996	Vaitzblit et al.	364/514A
5530557	June 1996	Asit et al.	358/342
5533021	July 1996	Branstad et al.	370/60.1
5534912	July 1996	Kostreski	348/6
5537408	July 1996	Branstad et al.	370/79
5539660	July 1996	Blair et al.	364/514C
5544313	August 1996	Shachnai et al.	395/200.01
5544327	August 1996	Dan et al.	395/250
5544345	August 1996	Carpenter et al.	395/477

h e b b g e e e f e ef b e

5544347	August 1996	Yanai et al.	395/489
5550577	August 1996	Verbiest et al.	348/7
5550982	August 1996	Long et al.	395/200.13
5553005	September 1996	Voeten et al.	364/514R
5557317	September 1996	Nishio et al.	348/7
5559949	September 1996	Reimer et al.	395/161
5574662	November 1996	Windrem et al.	364/514R
5583561	December 1996	Baker et al.	347/7
5586264	December 1996	Belknap et al.	395/200.08
5594863	January 1997	Stiles	395/182.13
5594910	January 1997	Filepp et al.	395/800
5603058	February 1997	Belknap et al.	395/855
5606359	February 1997	Youden et al.	348/7
5608865	March 1997	Midgely et al.	395/180
5611069	March 1997	Matoba	395/441
5625405	April 1997	DuLac et al.	348/7
5633810	May 1997	Mandal et al.	364/514
5633999	May 1997	Clowes et al.	395/182.04
5649093	July 1997	Hanko et al.	395/182.04
5737747	April 1998	Vishlitzky et al.	711/118
5742792	April 1998	Yanai et al.	395/489
5774668	June 1998	Choquier et al.	395/200.53
5809543	September 1998	Byers et al.	711/162
5812748	September 1998	Ohran et al.	395/182.02
5815146	September 1998	Youden et al.	345/327
5829046	October 1998	Tzelnic et al.	711/162
5829047	October 1998	Jacks et al.	711/162

FOREIGN PATENT DOCUMENTS

FOREIGN-PAT-NO	PUBN-DATE	COUNTRY	US-CL
0 061 570 A3	October 1982	EP	
633694 A1	January 1995	EP	
0 683 464 A2	November 1995	EP	
0 697 660 A1	February 1996	EP	
WO 93/16557	August 1993	WO	
WO 94/00816	January 1994	WO	
WO 95/10918	April 1995	WO	
WO 97/16023	May 1997	WO	

OTHER PUBLICATIONS

M(aurice) William Collins, "A Network File Storage System," IEEE Seventh Symposium on Mass Storage Systems, Nov. 4-7, 1985, Tuscon, AZ, p. 1-11, Los Alamos Nat. Lab. No. LA-UR-85-3183.
John H. Howard, "An Overview of the Andrew File System," USENIX Winter Conference, Feb. 9-12, 1988, Dallas, TX, p. 23-26.
John H. Howard et al., "Scale and Performance in a Distributed File System," ACM

Transactions on Computer Systems, vol. 6, No. 1, Feb. 1988, p. 51-81.
David C. Steere et al., "Efficient User-Level File Cache Management on the Sun Vnode Interface," USENIX Summer Conference, Jun. 11-15, 1990, Anaheim, CA, p. 325-331.
Matt Blaze et al., "Long-Term Caching Strategies for Very Large Distributed File Systems," USENIX, Summer '91, Nashville, TN, pp. 3-15.
Thomas W. Page, Jr., et al., "Management of Replicated Volume Location Data in the Ficus Replicated File System," USENIX, Summer '91, Nashville, TN, p. 17-29.
Storage Computer Corporation, "High Performance, Fault Tolerant Disk Array Platform For File Servers And Computer Systems," 1991, Nashua, NH, 12 pages.
France Telecom, "Telesauvegarde," 26 page paper dated Nov. 2, 1994 about work based on France Telecom patent application "Dispositif et Proceed de Sauvegaude a Distance de Donnees Numeriques" [Method and Apparatus for Safeguarding at a Distance Numeric Data], Institute National de la Propiete Industrielle, France, Appln. No. 93.12771, Oct. 28, 1994 (filed Oct. 21, 1993), and partial English translation (15 pages).
Krishnan Natarajan, "Video Servers Take Root," IEEE Spectrum, Apr. 1995, IEEE, New York, NY, p. 66-69.
K. K. Ramakrishnan et al., "Operating system support for a video-on-demand file service," Multimedia Systems (1995) 3:53-65.
Ralf Steinmetz, "Analyzing the Multimedia Operating System," IEEE MultiMedia, Spring 1995, p. 68-84.
Audrey Chou, "EMC, Computer-Storage Leader, Still Hears Footsteps," The Wall Street Journal, Aug. 9, 1995, Dow Jones & Co., Princeton, N.J.
Pardhu Vadlamudi, "EMC Hunts for Solution to Bottlenecks," InfoWorld, Apr. 15, 1996, #1590, San Mateo, CA 94402.
Michael Goldberg, "EMC to Pump Data Over Networks," Computerworld, Apr. 15, 1996.
"EMC Moves Into Servers," Broadcasting Cable, Apr. 15, 1996.
"Symmetrix Model 55XX Product Manual, P/N 200-810-550 Rev D," EMC Corporation, Hopkinton, Mass., May 1994, pp. 1-236.
"NFS: Network File System Protocol Specification," RFC 1094, Sun Microsystems, Inc., Mar. 1989, pp. 1-27.
J. Case, M. Fedor, M. Schoffstall, J. Davin, "A Simple Network Management Protocol (SNMP)," May 1990, MIT Laboratory for Computer Science, Cambridge, Mass., pp. 1-35.

Rangen PV, Vin HM, "Designing file systems for digital video and audio," Proceedings of the 13th ACM Symposium on Operating Systems Principles, Monterey, Calif., 1991, pp. 81-94.
Vin HM, Rangan PV, (1993) "Designing a multiuser HDTV storage server," IEEE Journal on Selected Areas in Communication, vol. 11, No. 1, Jan. 1993, pp. 153-164.
Anderson DP, Osawa Y, Govindan R, "A file system for continuous media," ACM Transactions on Computer Systems, vol. 10., No. 4, Nov. 1992, pp. 311-337.
Federighi C, "A Distributed Hierarchical Storage Manager for a Video-on-Demand System," Department of Electrical Engr. and Computer Science, University of California, Berkeley, California, Dec. 1993.
Haskins R, "The shark continuous-media file server," Proceedings IEEE COMPCON 93, San Francisco, Calif., 1993, pp. 12-15.
Little TD, Rhanger G, Folz, RJ, Gibbon JF, Reeve FW, Schelleng DH, Venkatesh D, "A digital on-demand video service supporting content based queries," Proceedings of ACM Multimedia 93, Anaheim, Calif., Aug. 1-6, 1993, pp. 427-436.
Lougher, P, Sheperd, D. "The design of a storage server for continuous media," The Computer Journal, vol. 36, No. 1, 1993, pp. 32-42.
Rangan PV, Vin HM, Ramanathan S, "Designing an on-demand multimedia service," IEEE Communications Magazine, vol. 30, No. 7, Jul. 1992, pp. 56-64.
Sincoskie, WD, "System architecture for a large scale video on demand service," Computer Networks and ISDN Systems, vol. 22, No. 2, Sep. 1991, pp. 155-162.
Tobagi FA, Pang J, "StarWorks (Trademark)--A video applications server," Proceedings, IEEE COMPCON 93, San Francisco, Calif., 1993, pp. 4-11.
Vaitzblit L, "The design and implementation of a high bandwidth file service for continuous media," Master's Thesis, Massachusetts Institute of Technology,

Cambridge, Mass., Nov. 4, 1991.
D.L. Burkes & R.K. Treiber,"Design Approaches for Real-Time Transaction Processing Remote Site Recovery," Computer Society International Conference (COMPCON), Spring Meeting, Los Alamitos, Feb. 26-Mar. 2, 1990, No., Conf. 35, Feb. 23, 1990, Institute of Electrical and Electronics Engineers, New York, N.Y., pp. 568-572.
SFT Netware 286 Maintenance, #100-313-001, 100/Rev1.00, Novel Incorporated, Provo, Utah, Nov. 1987.
SFT Netware 286 Installation,#100-312-001, 100/Rev1.00, Novel Incorporated, Provo, Utah, Nov. 1987.
SFT Netware 286 Installation Supplement, #100-000225-001, 26/Rev1.02, Novel Incorporated, Provo, Utah, Jun. 1986.

ART-UNIT: 275

PRIMARY-EXAMINER: Beausoliel, Jr.; Robert W.

ASSISTANT-EXAMINER: Iqbal; Nadeem

ATTY-AGENT-FIRM: Arnold White & Durkee

ABSTRACT:

A file server includes stream server computers linking a cached disk array storage subsystem to a data network, and at least two controller servers for receiving requests for file access from network clients. At any given time one of the controller servers is active and another is inactive in servicing client requests. The active controller server selects one of the stream servers to service each request. A controller server failover mechanism is provided for recovering from a failure of the active controller server, and a stream server failover mechanism is provided for recovering from a failure of a stream server. The inactive controller server becomes active when it fails to receive a signal periodically transmitted by the active controller server. The active controller server begins stream server failover when it fails to receive a signal periodically transmitted by each stream server. To resume automatically an interrupted task, the tasks are organized as a series of transactions, and each transaction includes operations which can be duplicated without substantial disruption. The active controller server commits results of each transaction to memory of the cached disk array. Before becoming active, the inactive controller recovers the committed state of the interrupted tasks from the cached disk array.

45 Claims, 43 Drawing figures

☐ 6. Document ID: US 5974503 A

L11: Entry 6 of 16 File: USPT Oct 26, 1999

US-PAT-NO: 5974503
DOCUMENT-IDENTIFIER: US 5974503 A

TITLE: Storage and access of continuous media files indexed as lists of raid stripe sets associated with file names

DATE-ISSUED: October 26, 1999

h e b b g e e e f e ef b e

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Venkatesh; Dinesh	Brookline	MA		
Forecast; John	Newton	MA		
Duso; Wayne W.	Shrewsbury	MA		

ASSIGNEE-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY	TYPE CODE
EMC Corporation	Hopkinton	MA			02

APPL-NO: 08/ 851509 [PALM]
DATE FILED: May 5, 1997

PARENT-CASE:
RELATED APPLICATIONS The present application is a continuation of provisional application Serial No. 60/044,948 filed Apr. 25, 1997 by Dinesh Venkatesh, Wayne W. Duso, John Forecast, Uday Gupta, Uresh K. Vahalia, and Dennis P. J. Ting, entitled "Raid Striping, Client-Server Protocols, and Failover Services for a Video File Server."

INT-CL: [06] G06 F 12/00

US-CL-ISSUED: 711/114; 395/182.04, 395/182.05
US-CL-CURRENT: 711/114; 714/6, 714/7

FIELD-OF-SEARCH: 711/114, 371/51.1, 371/40.4, 395/182.04, 395/182.05

PRIOR-ART-DISCLOSED:

U.S. PATENT DOCUMENTS

PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL
4608688	August 1986	Hansen et al.	371/11
4755928	July 1988	Johnson et al.	364/200
4761785	August 1988	Clark et al.	371/51.1
5072378	December 1991	Manka	395/575
5148432	September 1992	Gordon et al.	371/10.1
5175837	December 1992	Arnold et al.	395/425
5206939	April 1993	Yani et al.	395/400
5208665	May 1993	McCalley et al.	358/862
5208813	May 1993	Stallmo et al.	371/10.1
5218695	June 1993	Noveck et al.	395/600
5235601	August 1993	Stallmo et al.	371/40.4
5255270	October 1993	Yanai et al.	371/10.2
5263145	November 1993	Brady et al.	395/425
5269011	December 1993	Yanai et al.	395/425
5274799	December 1993	Brant et al.	385/575
5276860	January 1994	Fortier et al.	395/575
5276867	January 1994	Kenley et al.	395/600
5335352	August 1994	Yanai et al.	395/800

h e b b g e e e f e ef b e

5341493	August 1994	Yanai et al.	395/425
5367698	November 1994	Webber et al.	395/800
5371532	December 1994	Gelman et al.	348/7
5381539	January 1995	Yanai et al.	395/425
5408465	April 1995	Gusella et al.	370/17
5410343	April 1995	Coddington et al.	348/7
5414455	May 1995	Hooper et al.	348/7
5442390	August 1995	Hooper et al.	348/7
5442749	August 1995	Northcutt et al.	395/200.09
5442771	August 1995	Filepp et al.	395/200.08
5477263	December 1995	O'Callaghan et al.	348/7
5508733	April 1996	Kassatly	348/13
5519435	May 1996	Anderson	348/8
5528282	June 1996	Voeten et al.	348/7
5528513	June 1996	Vaitzblit et al.	364/514A
5530557	June 1996	Asit et al.	358/342
5533021	July 1996	Branstad et al.	370/60.1
5534912	July 1996	Kostreski	348/6
5537408	July 1996	Branstad et al.	370/79
5537534	July 1996	Voigt et al.	395/182.04
5539660	July 1996	Blair et al.	364/514C
5544313	August 1996	Shachnai et al.	395/200.01
5544327	August 1996	Dan et al.	395/250
5544345	August 1996	Carpenter et al.	395/477
5544347	August 1996	Yanai et al.	395/489
5550577	August 1996	Verbiest et al.	348/7
5550982	August 1996	Long et al.	395/200.13
5553005	September 1996	Voeten et al.	364/514R
5557317	September 1996	Nishio et al.	348/7
5559949	September 1996	Reimer et al.	395/161
5572660	November 1996	Jones	395/182.04
5574662	November 1996	Windrem et al.	364/514R
5579475	November 1996	Blaum et al.	395/182.05
5583561	December 1996	Baker et al.	347/7
5586264	December 1996	Belknap et al.	395/200.08
5594910	January 1997	Filepp et al.	395/800
5603058	February 1997	Belknap et al.	395/855
5606359	February 1997	Youden et al.	348/7
5625405	April 1997	DuLac et al.	348/7
5633810	May 1997	Mandal et al.	364/514
5657439	August 1997	Jones et al.	395/182.05
5812753	September 1998	Chiariotti	395/182.04

FOREIGN PATENT DOCUMENTS

FOREIGN-PAT-NO	PUBN-DATE	COUNTRY	US-CL

h e b b g e e e f e ef b e

0 061 570 A3	October 1982	EP
633694 A1	January 1995	EP
0 683 464 A2	November 1995	EP
0 697 660 A1	February 1996	EP
WO 93/16557	August 1993	WO
WO 94/00816	January 1994	WO
WO 95/10918	April 1995	WO
WO 97/16023	May 1997	WO

OTHER PUBLICATIONS

Dishon et al., "A Highly Available Storage System Using the Checksum Method," Seventeenth International Symposium on Fault-Tolerant Computing, Jul. 6-8, 1987, Pittsburg, Penn., IEEE Computer Society, IEEE, New York, N.Y., pp. 178-181.
Martin E. Schulze, "Considerations in the Design of a RAID Prototype," Computer Science Division, EECS, University of California at Berkeley, CA, Aug. 25, 1988.
Katz et al., "Disk System Architectures for High Performance Computing," Computer Science Division, EECS, University of California at Berkeley, CA, Mar. 1989.
Gray et al., "Parity Striping of Disc Arrays: Low-Cost Reliable Storage with Acceptable Throughput," Technical Report 90.2, Tandem Computers, Cupertino, CA., Jan. 1990.
Edward K. Lee, "Software and Performance Issues in the Implementation of a RAID Prototype," Computer Science Division, EECS, University of California at Berkeley, CA, Mar. 1989.
Sincoskie, WD, "System architecture for a large scale video on demand service," Computer Networks and ISDN Systems, vol. 22, No. 2, Sep. 1991, pp. 155-162.
Tobagi FA, Pang J, "StarWorks (Trademark)--A video applications server," Proceedings, IEEE COMPCON 93, San Francisco, Calif., 1993, pp. 4-11.
Vaitzblit L, "The design and implementation of a high bandwidth file service for continuous media," Master's Thesis, Massachusetts Institute of Technology, Cambridge, Mass., Nov. 4, 1991.
Patterson et al., "A Case for Redundant Arrays Of Inexpensive Disks (RAID)," Report No. UCB/CSD 87/391, Computer Science Division (EECS), University of California, Berkeley, California, Dec. 1987, pp. 1-24.
Patterson et al., "Introduction to Redundant Arrays of Inexpensive Disks (RAID)," COMPCON 89 Proceedings, Feb. 27-Mar. 3, 1989, IEEE Computer Society, pp. 112-117.
Ousterhout et al., "Beating the I/O Bottleneck: A Case for Log-Structured File Systems," Operating Systems Review, vol. 23, No. 1, ACM Press, Jan., 1989, pp. 11-28.
Douglis et al., "Log Structured File Systems," COMPCON 89 Proceedings, Feb. 27-Mar. 3, 1989, IEEE Computer Society, pp. 124-129.
Rosemblum et al., "The Design and Implementation of a Log-Structured File System," ACM Transactions on Computer Systems, vol. 1, Feb. 1992, pp. 26-52.
M(aurice) William Collins, "A Network File Storage System," IEEE Seventh Symposium on Mass Storage Systems, Nov. 4-7, 1985, Tuscon, AZ, pp. 1-11, Los Alamos Nat. Lab. No. LA-UR-85-3183.
John H. Howard, "An Overview of the Andrew File System," USENIX Winter Conference, Feb. 9-12, 1988, Dallas, TX, p. 23-26.
John H. Howard et al., "Scale and Performance in a Distributed File System," ACM Transactions on Computer Systems, vol. 6, No. 1, Feb. 1988, p. 51-81.
David C. Steere et al., "Efficient User-Level File Cache Management on the Sun Vnode Interface," USENIX Summer Conference, Jun. 11-15, 1990, Anaheim, CA, p. 325-331.
Matt Blaze et al., "Long-Term Caching Strategies for Very Large Distributed File Systems," USENIX, Summer '91, Nashville, TN, p. 3-15.
Thomas W. Page, Jr., et al., "Management of Replicated Volume Location Data in the Ficus Replicated File System," USENIX, Summer '91, Nashville, TN, p. 17-29.

Storage Computer Corporation, "High Performance, Fault Tolerant Disk Array Platform For File Servers And Computer Systems," 1991, Nashua, NH 12 pages.
France Telecom, "Telesauvegarde," 26 page paper dated Nov. 2, 1994 about work based on France Telecom patent application "Dispositif et Proceed de Sauvegaude a Distance de Donnees Numeriques" [Method and Apparatus for Safeguarding at a Distance Numeric Data], Institute National de la Propiete Industrielle, France, Appln. No. 93.12771, Oct. 28, 1994 (filed Oct. 21, 1993), and partial English translation (15 pages).
Krishnan Natarajan, "Video Servers Take Root," IEEE Spectrum, Apr. 1995, IEEE, New York, NY, pp. 66-69.
K. K. Ramakrishnan et al., "Operating system support for a video-on-demand file service," Multimedia Systems (1995) 3:53-65.
Ralf Steinmetz, "Analyzing the Multimedia Operating System," IEEE MultiMedia, Spring 1995, pp. 68-84.
Audrey Chou, "EMC, Computer-Storage Leader, Still Hears Footsteps," The Wall Street Journal, Aug. 9, 1995, Dow Jones & Co., Princeton, N.J.
Pardhu Vadlamudi, "EMC Hunts for Solution to Bottlenecks," InfoWorld, Apr. 15, 1996, #1590, San Mateo, CA 94402.
Michael Goldberg, "EMC to Pump Data Over Networks," Computerworld, Apr. 15, 1996.
"EMC Moves Into Servers," Broadcasting Cable, Apr. 15, 1996.
"Symmetrix Model 55XX Product Manual, P/N 200-810-550 Rev D," EMC Corporation, Hopkinton, Mass., May 1994, pp. 1-236.
"NFS: Network File System Protocol Specification," RFC 1094, Sun Microsystems, Inc., Mar. 1989, pp. 1-27.
J. Case, M. Fedor, M. Schoffstall, J. Davin, "A Simple Network Management Protocol (SNMP)," May 1990, MIT Laboratory for Computer Science, Cambridge Mass., pp. 1-35.
Rangen PV, Vin HM, "Designing file systems for digital video and audio," Proceedings of the 13th ACM Symposium on Operating Systems Principles, Monterey, Calif., 1991, pp. 81-94.
Vin HM, Rangan PV, (1993) "Designing a multiuser HDTV storage server," IEEE Journal on Selected Areas in Communications, vol. 11, No. 1, Jan. 1993, pp. 153-164.
Anderson DP, Osawa Y, Govindan R, "A file system for continuous media," ACM Transactions on Computer Systems, vol. 10., No. 4, Nov. 1992, pp. 311-337.
Federighi C, "A Distributed Hierarchical Storage Manager for a Video-on-Demand System," Department of Electrical Engr. and Computer Science, University of California, Berkeley, California, Dec. 1993.
Haskins R, "The shark continuous-media file server," Proceedings IEEE COMPCOM 93, San Francisco, Calif., 1993, pp. 12-15.
Little TD, Rhanger G, Folz RJ, Gibbon JF, Reeve FW, Schelleng DH, Venkatesh D, "A digital on-demand video service supporting content based queries," Proceedings of ACM Multimedia 93, Anaheim, Calif., Aug. 1-6, 1993, pp. 427-436.
Lougher, P. Sheperd, D. "The design of a storage server for continuous media," The Computer Journal, vol. 36, No. 1, 1993, pp. 32-42.
Rangan PV, Vin HM, Ramanathan S, "Designing an on-demand multimedia service," IEEE Communications Magazine, vol. 30, No. 7, Jul. 1992, pp. 56-64.

ART-UNIT: 272

PRIMARY-EXAMINER: Cabeca; John W.

ASSISTANT-EXAMINER: Moazzami; Nasser

ATTY-AGENT-FIRM: Arnold White & Durkee

ABSTRACT:

A continuous media file is comprised of stripe sets over disk drives in one or more RAID sets. In a preferred embodiment, the RAID set includes n disk drives. The data storage of each disk drive in the RAID set is partitioned into an integer number m

h e b b g e e e f e ef b e

of hyper-volumes, and the parity is stored in one hyper-volume of each of m disk drives in the RAID set. The stripe set includes a series of transfer units of data in respective ones of the disk drives. Each transfer unit includes an integer number j of data blocks, and each hyper-volume includes an integer number k of transfer units. Each stripe set includes (m)(n-1) transfer units of data. The transfer units of the RAID set are allocated for the storage of continuous media data in a right-to-left and then top-to-bottom order in which the transfer units appear in an m row by n column matrix in which the rows of the matrix represent parity groups of hyper-volumes in the disk drives and the columns of the matrix represent storage in the respective disk drives. At most one write access to each parity hyper-volume need be performed during write access to a stripe set. Parity changes for the data being written are accumulated in non-volatile memory, and written to the RAID set after completion of the writing of the data.

12 Claims, 42 Drawing figures

☐ 7. Document ID: US 5933603 A

L11: Entry 7 of 16 File: USPT Aug 3, 1999

US-PAT-NO: 5933603
DOCUMENT-IDENTIFIER: US 5933603 A

TITLE: Video file server maintaining sliding windows of a video data set in random access memories of stream server computers for immediate video-on-demand service beginning at any specified location

DATE-ISSUED: August 3, 1999

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Vahalia; Uresh K.	Newton	MA		
Forecast; John	Newton	MA		
Tzelnic; Percy	Concord	MA		

ASSIGNEE-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY	TYPE CODE
EMC Corporation	Hopkinton	MA			02

APPL-NO: 08/ 661053 [PALM]
DATE FILED: June 10, 1996

PARENT-CASE:
RELATED APPLICATIONS The present application is a divisional of provisional application Ser. No. 60/005,988 filed Oct. 27, 1995 by Percy Tzelnic et al., entitled "Video File Server," fully incorporated herein by reference. The detailed description in the present application has been updated to conform to the applicants' present implementation of their invention. The present application also contains a disclosure which is common with the following other divisional applications of Ser. No. 60/005,988: Percy Tzelnic et al., Ser. No. 08/661,152 filed Jun. 10, 1996, entitled "Video File Server Using an Integrated Cached Disk Array and Stream Server Computers"; Natan Vishlitzky et al., Ser. No. 08/661,185

h e b b g e e e f e ef b e

filed Jun. 10, 1996, entitled "Prefetching to Service Multiple Video Streams From an Integrated Cached Disk Array" and issued Apr. 7, 1998, as U.S. Pat. No. 5,737,747; and Percy Tzelnic et al., Ser. No. 08/661,187 filed Jun. 10, 1996, entitled "On-Line Tape Backup Using an Integrated Cached Disk Array and issued on Oct. 27, 1998, as U.S. Pat. No. 5,829,046"

INT-CL: [06] H04 N 7/173, H04 N 7/16, G06 F 13/14

US-CL-ISSUED: 395/200.55; 395/200.33, 395/200.47, 395/200.56, 395/200.49, 395/200.59, 711/100, 711/105
US-CL-CURRENT: 709/225; 711/100, 711/105

FIELD-OF-SEARCH: 395/200.01, 395/200.03, 395/200.08, 395/200.09, 395/200.06, 395/200.3, 395/200.31, 395/200.33, 395/200.36, 395/200.47, 395/200.48, 348/6-8, 348/12-13, 455/4.2, 455/5.1, 711/100, 711/105

PRIOR-ART-DISCLOSED:

U.S. PATENT DOCUMENTS

PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL
4608688	August 1986	Hansen et al.	395/182.04
4755928	July 1988	Johnson et al.	395/182.04
5175837	December 1992	Arnold et al.	711/152
5206939	April 1993	Yanai et al.	711/4
5208665	May 1993	McCalley et al.	348/12
5218695	June 1993	Noveck et al.	707/205
5255270	October 1993	Yanai et al.	371/10.2
5269011	December 1993	Yanai et al.	395/280
5276860	January 1994	Fortier et al.	395/182.04
5276867	January 1994	Kenley et al.	707/204
5335352	August 1994	Yanai et al.	395/871
5341493	August 1994	Yanai et al.	711/161
5367698	November 1994	Webber et al.	395/200.33
5371532	December 1994	Gelman et al.	348/7
5381539	January 1995	Yanai et al.	711/133
5408465	April 1995	Gusella et al.	370/231
5410343	April 1995	Coddington et al.	348/7
5414455	May 1995	Hooper et al.	348/7
5442390	August 1995	Hooper et al.	348/7
5442749	August 1995	Northcutt et al.	395/200.66
5442771	August 1995	Filepp et al.	395/200.49
5477263	December 1995	O'Callaghan et al.	348/7
5508732	April 1996	Bottomley et al.	348/7
5508733	April 1996	Kassatly	348/13
5519435	May 1996	Anderson	348/8
5528282	June 1996	Voeten et al.	348/7
5528513	June 1996	Vaitzblit et al.	395/673
5530557	June 1996	Asit et al.	386/125
5533021	July 1996	Branstad et al.	370/396

5534912	July 1996	Kostreski	348/6
5537408	July 1996	Branstad et al.	370/473
5539660	July 1996	Blair et al.	370/380
5544313	August 1996	Shachnai et al.	395/200.49
5544327	August 1996	Dan et al.	395/200.64
5544345	August 1996	Carpenter et al.	711/150
5544347	August 1996	Yanai et al.	711/162
5550577	August 1996	Verbiest et al.	348/7
5550982	August 1996	Long et al.	395/200.49
5553005	September 1996	Voeten et al.	711/112
5557317	September 1996	Nishio et al.	348/7
5559949	September 1996	Reimer et al.	707/3
5574662	November 1996	Windrem et al.	395/200.49
5583561	December 1996	Baker et al.	348/7
5586264	December 1996	Belknap et al.	395/200.49
5594910	January 1997	Filepp et al.	395/800.28
5603058	February 1997	Belknap et al.	395/855
5606359	February 1997	Youden et al.	348/7
5610653	March 1997	Abecassis	348/170
5625405	April 1997	DuLac et al.	348/7
5633810	May 1997	Mandal et al.	370/431

FOREIGN PATENT DOCUMENTS

FOREIGN-PAT-NO	PUBN-DATE	COUNTRY	US-CL
0 061 570 A3	October 1982	EP	
633694	January 1995	EP	
0 683 464 A2	November 1995	EP	
0 697 660 A1	February 1996	EP	
0 701 372 A1	March 1996	EP	
WO93/16557	August 1993	WO	
WO 94/00816	January 1994	WO	
WO95/10918	April 1995	WO	

OTHER PUBLICATIONS

International Search Report for PCT/US 96/17156, European Patent Office, Rijswijk, Feb. 21, 1997, 3 pages.
Rangen PV, Vin HM, "Designing file systems for digital video and audio," Proceedings of the 13th ACM Symposium on Operating Systems Principles, Monterey, Calif., 1991, pp. 81-94.
Vin HM, Rangan PV, (1993), "Designing a multiuser HDTV storage server," IEEE Journal on Selected Areas in Communication, vol. 11, No. 1, Jan. 1993, pp. 153-164.

Anderson DP, Osawa Y, Govindan R, "A file system for continuous media," ACM Transactions on Computer Systems, vol. 10., No. 4, Nov. 1992, pp. 311-337.
Federighi C, "A Distributed Hierarchical Storage Manager for a Video-on-Demand System," Department of Electrical Engr. and Computer Science, University of California, Berkeley, California, Dec. 1993.
Haskin R, The shark continuous-media file server. Proceedings IEEE COMPCON 93, San

Francisco, Calif., 1993, pp. 12-15.
Little TD, Rhanger G, Folz RJ, Gibbon JF, Reeve FW, Schelleng DH, Venkatesh D, "A digital on-demand video service supporting content based queries," Proceedings of ACM Multimedia 93, Anaheim, Calif., Aug. 1-6, 1993, pp. 427-436.
Lougher, P, Sheperd, D. "The design of a storage server for continuous media," The Computer Journal, vol. 36, No. 1, 1993, pp. 32-42.
Rangan PV, Vin HM, Ramanathan S, "Designing an on-demand multimedia service," IEEE Communications Magazine, vol. 30, No. 7, Jul. 1992, pp. 56-64.
Sincoskie, WD, "System architecture for a large scale video on demand service,"Computer Networks and ISDN Systems, vol. 22, No. 2, Sep. 1991, pp. 155-162.
Tobagi FA, Pang J, "StarWorks (Trademark)--A video applications server," Proceedings, IEEE COMPCON 93, San Francisco, Calif., 1993, pp. 4-11.
Vaitzblit L, "The design and implementation of a high band-width file service for continuous media," Master's Thesis, Massachusetts Institute of Technology, Cambridge, Mass., Nov. 4, 1991.
M(aurice) William Collins, "A Network File Storage System," IEEE Seventh Symposium on Mass Storage Systems, Nov. 4-7, 1985, Tuscon, AZ, pp. 1-11, Los Alamos Nat. Lab. No. LA-UR-85-3183.
John H. Howard, "An Overview of the Andrew File System," USENIX Winter Conference, Feb. 9-12, 1988, Dallas, TX, pp. 23-26.
John H. Howard et al., "Scale and Performance in a Distributed File System," ACM Transactions on Computer Systems, vol. 6, No. 1, Feb. 1988, pp. 51-81.
David C. Steere et al., "Efficient User-Level File Cache Management on the Sun Vnode Interface," USENIX Summer Conference, Jun. 11-15, 1990, Anaheim, CA, pp. 325-331.
Matt Blaze et al., "Long-Term Caching Strategies for Very Large Distributed File Systems," USENIX, Summer '91, Nashville, TN, pp. 3-15.
Thomas W. Page, Jr., et al., "Management of Replicated Volume Location Data in the Ficus Replicated File System," USENIX, Summer '91, Nashville, TN, pp. 17-29.
Storage Computer Corporation, "High Performance, Fault Tolerant Disk Array Platform For File Servers And Computer Systems," 1991, Nashua, NH, 12 pages.
France Telecom, "Telesauvegarde," 26 page paper dated Nov. 2, 1994 about work based on France Telecom patent application "Dispositif et Proceed de Sauvegaude a Distance de Donnees Numeriques" [Method and Apparatus for Safeguarding at a Distance Numeric Data], Institute National de la Propiete Industrielle, France, Appln. No. 93.12771, Oct. 28, 1994 (filed Oct. 21, 1993), and partial English translation (15 pages).
Krishnan Natarajan, "Video Servers Take Root,"IEEE Spectrum, Apr. 1995, IEEE, New York, NY, pp. 66-69.
K. K. Ramakrishnan et al., "Operating system support for a video-on-demand file service," Multimedia Systems (1995) 3:53-65.
Ralf Steinmetz, "Analyzing the Multimedia Operating System," IEEE Multimedia, Spring 1995, pp. 68-84.
Audrey Chou, "EMC, Computer-Storage Leader, Still Hears Footsteps," The Wall Street Journal, Aug. 9, 1995, Dow Jones & Co., Princeton, N.J.
Pardhu Vadlamudi, "EMC Hunts for Solutions to Bottlenecks," InfoWorld, Apr. 15, 1996, #1590, San Mateo, CA 94402.
Michael Goldberg, "EMC to Pump Data Over Networks," Computerworld, Apr. 15, 1996.
"EMC Moves Into Servers," Broadcasting Cable, Apr. 15, 1996.
"Symmetrix Model 55XX Product Manual, P/N 200-810-550 Rev D," EMC Corporation, Hopkinton, Mass., May 1994, pp. 1-236.
"NFS: Network File System Protocol Specification," RFC 1094, Sun Microsystems, Inc., Mar. 1989, pp. 1-27.
J. Case, M. Fedor, M. Schoffstall, J. Davin, "A Simple Network Management Protocol (SNMP)," May 1990, MIT Laboratory for Computer Science, Cambridge, Mass., pp. 1-35.

ART-UNIT: 278

h e b b g e e e f e ef b e

PRIMARY-EXAMINER: Lim; Krisna

ASSISTANT-EXAMINER: Barot; Bharat

ATTY-AGENT-FIRM: Arnold White & Durkee

ABSTRACT:

A video file server includes an integrated cached disk array storage subsystem and a plurality of stream server computers linking the cached disk storage subsystem to a data network for the transfer of video data streams. The video file server further includes a server controller for applying an admission control policy to client requests and assigning stream servers to service the client requests. The stream servers include a real-time scheduler for scheduling isochronous tasks, and supports at least one industry standard network file access protocol and one file access protocol for continuous media file access. The cached disk storage subsystem is responsive to video prefetch commands, and the data specified for a prefetch command for a process are retained in an allocated portion of the cache memory from the time that the cached disk storage subsystem has responded to the prefetch command to the time that the cached disk storage subsystem responds to a fetch command specifying the data for the process. The time between prefetching and fetching is selected based on available disk and cache resources. The video file server provides video-on-demand service by maintaining and dynamically allocating sliding windows of video data in the random access memories of the stream server computers.

21 Claims, 20 Drawing figures

☐ 8. Document ID: US 5892915 A

L11: Entry 8 of 16 File: USPT Apr 6, 1999

US-PAT-NO: 5892915
DOCUMENT-IDENTIFIER: US 5892915 A

TITLE: System having client sending edit commands to server during transmission of continuous media from one clip in play list for editing the play list

DATE-ISSUED: April 6, 1999

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Duso; Wayne W.	Shrewsbury	MA		
Forecast; John	Newton	MA		

ASSIGNEE-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY	TYPE CODE
EMC Corporation	Hopkinton	MA			02

APPL-NO: 08/ 851560 [PALM]
DATE FILED: May 5, 1997

h e b b g e e e f e ef b e

PARENT-CASE:
RELATED APPLICATIONS The present application is a continuation of provisional application Ser. No. 60/044,948 filed Apr. 25, 1997 by Dinesh Venkatesh, Wayne W. Duso, John Forecast, Uday Gupta, Uresh K. Vahalia, and Dennis P. J. Ting, entitled "Raid Striping, Client-Server Protocols, and Failover Services for a Video File Server."

INT-CL: [06] G06 F 13/00

US-CL-ISSUED: 395/200.49; 348/7
US-CL-CURRENT: 709/219; 725/116, 725/93

FIELD-OF-SEARCH: 395/200.33, 395/200.49, 395/200.57, 395/200.56, 395/200.59, 348/7, 348/12, 348/13, 348/552, 348/906

PRIOR-ART-DISCLOSED:

U.S. PATENT DOCUMENTS

PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL
4608688	August 1986	Hansen et al.	371/11
4755928	July 1988	Johnson et al.	364/200
5109482	April 1992	Bohrman	345/328
5175837	December 1992	Arnold et al.	395/425
5206939	April 1993	Yanai et al.	395/400
5208665	May 1993	McCalley et al.	358/862
5218672	June 1993	Morgan et al.	345/501
5218695	June 1993	Noveck et al.	395/600
5255270	October 1993	Yanai et al.	371/10.2
5267092	November 1993	Kizu et al.	386/55
5269011	December 1993	Yanai et al.	395/425
5276860	January 1994	Fortier et al.	395/575
5276867	January 1994	Kenley et al.	395/600
5335352	August 1994	Yanai et al.	395/800
5341493	August 1994	Yanai et al.	395/425
5367698	November 1994	Webber et al.	395/800
5371532	December 1994	Gelman et al.	348/7
5381539	January 1995	Yanai et al.	395/425
5408465	April 1995	Gusella et al.	370/17
5410343	April 1995	Coddington et al.	348/7
5414455	May 1995	Hooper et al.	348/7
5442390	August 1995	Hooper et al.	348/7
5442771	August 1995	Filepp et al.	395/200.08
5469270	November 1995	Yamamoto	386/55
5477263	December 1995	O'Callaghan et al.	348/7
5508732	April 1996	Bottomley et al.	348/7
5508733	April 1996	Kassatly	348/13
5528282	June 1996	Voeten et al.	348/7
5528513	June 1996	Vaitzblit et al.	364/514A

h e b b g e e e f e ef b e

5530557	June 1996	Asit et al.	358/342
5533021	July 1996	Branstad et al.	370/60.1
5534912	July 1996	Kostreski	348/6
5537408	July 1996	Branstad et al.	370/79
5539660	July 1996	Blair et al.	364/514C
5544313	August 1996	Shachnai et al.	395/200.01
5544327	August 1996	Dan et al.	395/250
5544345	August 1996	Carpenter et al.	395/477
5544347	August 1996	Yanai et al.	395/489
5550577	August 1996	Verbiest et al.	348/7
5553005	September 1996	Voeten et al.	364/514R
5553281	September 1996	Brown et al.	707/104
5557317	September 1996	Nishio et al.	348/7
5559949	September 1996	Reimer et al.	395/161
5583561	December 1996	Baker et al.	347/7
5584006	December 1996	Reber et al.	711/100
5586264	December 1996	Belknap et al.	395/200.08
5594910	January 1997	Filepp et al.	395/800
5603058	February 1997	Belknap et al.	395/855
5606359	February 1997	Youden et al.	348/7
5640320	June 1997	Jackson et al.	364/192
5687332	November 1997	Kurahashi et al.	345/335
5732219	March 1998	Blumer et al.	395/200.57
5799150	August 1998	Hamilton et al.	395/200.33

FOREIGN PATENT DOCUMENTS

FOREIGN-PAT-NO	PUBN-DATE	COUNTRY	US-CL
0 061 570 A3	October 1982	EP	
633694 A1	January 1995	EP	
0 683 464 A2	November 1995	EP	
0 697 660 A1	February 1996	EP	
0 701 372 A1	March 1996	EP	
WO 93/16557	August 1993	WO	
WO 94/00816	January 1994	WO	
WO 95/10918	April 1995	WO	
WO 97/10623	May 1997	WO	

OTHER PUBLICATIONS

M(aurice) William Collins, "A Network File Storage System," IEEE Seventh Symposium on Mass Storage Systems, Nov. 4-7, 1985, Tuscon, AZ, pp. 1-11, Los Alamos Nat. Lab. No. LA-UR-85-3183.
John H. Howard, "An Overview of the Andrew File System," USENIX Winter Conference, Feb. 9-12, 1988, Dallas, TX, pp. 23-26.
John H. Howard et al., "Scale and Performance in a Distributed File System," ACM Transactions on Computer Systems, vol. 6, No. 1, Feb. 1988, pp. 51-81.
David C. Steere et al., "Efficient User-Level File Cache Management on the Sun Vnode Interface," USENIX Summer Conference, Jun. 11-15, 1990, Anaheim, CA, pp. 325-

h e b b g e e e f e ef b e

331.
Matt Blaze et al., "Long-Term Caching Strategies for Very Large Distributed File Systems," USENIX, Summer '91, Nashville, TN, pp. 3-15.
Thomas W. Page, Jr., et al., "Management of Replicated Volume Location Data in the Ficus Replicated File System," USENIX, Summer '91, Nashville, TN, pp. 17-29.
Storage Computer Corporation, "High Performance, Fault Tolerant Disk Array Platform For File Servers And Computer Systems," 1991, Nashua, NH, 12 pages.
France Telecom, "Telesauvegarde," 26 page paper dated Nov. 2, 1994 about work based on France Telecom patent application Dispositif et Proceed de Sauvegaude a Distance de Donnees Numeriques [Method and Apparatus for Safeguarding at a Distance Numeric Data], Institute National de la Propiete Industrielle, France, Appln. No. 93.12771, Oct. 28, 1994 (filed Oct. 21, 1993), and partial English translation (15 pages).
Krishnan Natarajan, "Video Servers Take Root," IEEE Spectrum, Apr. 1995, IEEE, New York, NY, pp. 66-69.
K. K. Ramakrishnan et al., "Operating system support for a video-on-demand file service," Multimedia Systems (1995) 3:53-65.
Ralf Steinmetz, "Analyzing the Multimedia Operating System," IEEE MultiMedia, Spring 1995, pp. 68-84.
Audrey Chou, "EMC, Computer-Storage Leader, Still Hears Footsteps," The Wall Street Journal, Aug. 9, 1995, Dow Jones & Co., Princeton, N.J.
Pardhu Vadlamudi, "EMC Hunts for Solution to Bottlenecks," InfoWorld, Apr. 15, 1996, #1590, San Mateo, CA 94402.
Michael Goldberg, "EMC to Pump Data Over Networks," Computerworld, Apr. 15, 1996.
"EMC Moves Into Servers," Broadcasting Cable, Apr. 15, 1996.
"Symmetrix Model 55XX Product Manual, P/N 200-810-550 Rev D," EMC Corporation, Hopkinton, Mass., May 1994, pp.1-236.
"NFS: Network File System Protocol Specification ," RFC 1094, Sun Microsystems, Inc., Mar. 1989, pp. 1-27.
J. Case, M. Fedor, M. Schoffstall, J. Davin, "A Simple Network Management Protocol (SNMP)," May 1990, MIT Laboratory for Computer Science, Cambridge, Mass., pp. 1-35.

Rangen PV, Vin HM, "Designing file systems for digital video and audio," Proceedings of the 13th ACM Symposium on Operating Systems Principles, Monterey, Calif., 1991, pp. 81-94.
Vin HM, Rangan PV, (1993) "Designing a multiuser HDTV storage server," IEEE Journal on Selected Areas in Communication, vol. 11, No. 1, Jan. 1993, pp. 153-164.
Anderson DP, Osawa Y, Govindan R, "A file system for continuous media," ACM Transactions on Computer Systems, vol. 10., No. 4, Nov. 1992, pp. 311-337.
Federighi C, "A Distributed Hierarchical Storage Manager for a Video-on-Demand System," Department of Electrical Engr. and Computer Science, University of California, Berkeley, California, Dec. 1993.
Haskins R, "The shark continuous-media file server," Proceedings IEEE COMPCON 93, San Francisco, Calif., 1993, pp. 12-15.
Little TD, Rhanger G, Folz RJ, Gibbon JF, Reeve FW, Schelleng DH, Venkatesh D, "A digital on-demand video service supporting content based queries," Proceedings of ACM Multimedia 93, Anaheim, Calif., Aug. 1-6, 1993, pp. 427-436.
Lougher, P. Sheperd, D. "The design of a storage server for continuous media," The Computer Journal, vol. 36, No. 1, 1993, pp. 32-42.
Rangan PV, Vin HM, Ramanathan S, "Designing an on-demand multimedia service," IEEE Communications Magazine, vol. 30, No. 7, Jul. 1992, pp. 56-64.
Sincoskie, WD, "System architecture for a large scale video on demand service," Computer Networks and ISDN Systems, vol. 22, No. 2, Sep. 1991, pp. 155-162.
Tobagi FA, Pang J, "StarWorks (Trademark)--A video applications server," Proceedings, IEEE COMPCON 93, San Francisco, Calif., 1993, pp. 4-11.
Vaitzblit L, "The design and implementation of a high bandwidth file service for continuous media," Master's Thesis, Massachusetts Institute of Technology, Cambridge, Mass., Nov. 4, 1991.

ART-UNIT: 277

h e b b g e e e f e ef b e

PRIMARY-EXAMINER: Meky; Moustafa M.

ATTY-AGENT-FIRM: Arnold White & Durkee

ABSTRACT:

A protocol and interface provides continuous play over multiple clips for extended periods of time, allows a play-list to be edited dynamically after being given to the video server and during playback of clips in the play-list, allows some notion of "current time" to be used during the streaming of continuous media data, and supports features of the "Louth Automation" video disk communications protocol. Preferably, the client application first creates a session with a play-list containing a fixed number of entries; the number should be as small as possible consistent with the client's requirements. The client edits this play-list by appending the first few clips and then starts the session playing. Each time transmission of video data of a clip is completed, the clip is removed from the head of the play-list, all other clips are moved down, and a callback is issued to the client with the current, updated, play-list. A callback is also issued with the updated play-list to acknowledge each edit command. Preferably, there is a limit as to how close to air-time a clip normally may be deleted or new material inserted, in order to ensure continuity of transmission of the video stream of each clip. To allow live break-ins or other "emergency" operations, however, the session may be paused and later resumed and subsequent clips may be "trimmed" to reduce their play times to recover the time lost to the break-in.

63 Claims, 43 Drawing figures

☐ 9. Document ID: US 5829046 A

L11: Entry 9 of 16 File: USPT Oct 27, 1998

US-PAT-NO: 5829046
DOCUMENT-IDENTIFIER: US 5829046 A
**** See image for Certificate of Correction ****

TITLE: On-line tape backup using an integrated cached disk array

DATE-ISSUED: October 27, 1998

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Tzelnic; Percy	Concord	MA		
Dunham; Scott R.	Billerica	MA		

ASSIGNEE-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY	TYPE CODE
EMC Corporation	Hopkinton	MA			02

APPL-NO: 08/ 661187 [PALM]
DATE FILED: June 10, 1996

INT-CL: [06] G06 F 12/14

h e b b g e e e f e ef b e

US-CL-ISSUED: 711/162; 711/161, 711/156, 711/118, 711/111, 711/113, 711/100
US-CL-CURRENT: 711/162; 711/100, 711/111, 711/113, 711/118, 711/156, 711/161

FIELD-OF-SEARCH: 395/200.09, 395/200.13, 348/8, 348/7, 364/514R, 364/514, 711/162, 711/161

PRIOR-ART-DISCLOSED:

U.S. PATENT DOCUMENTS

PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL
5208665	May 1993	McCalley et al.	358/862
5218695	June 1993	Noveck et al.	395/600
5276860	January 1994	Fortier et al.	395/575
5276867	January 1994	Kenley et al.	395/600
5367698	November 1994	Webber et al.	395/800
5371532	December 1994	Gelman et al.	348/7
5442749	August 1995	Northcutt et al.	395/200.09
5442771	August 1995	Filepp et al.	395/200.08
5504873	April 1996	Martin et al.	395/438
5519435	May 1996	Anderson	348/8
5528513	June 1996	Vaitzblit et al.	364/514A
5550982	August 1996	Long et al.	395/200.13
5551025	August 1996	O'Reilley et al.	396/600
5574662	November 1996	Windrem et al.	364/514R
5586264	December 1996	Bleknap et al.	395/200.08
5590320	December 1996	Maxey	395/619
5594910	January 1997	Filepp et al.	395/800
5603058	February 1997	Belknap et al.	395/855
5606359	February 1997	Youden et al.	348/7
5625405	April 1997	DuLac et al.	348/7
5633810	May 1997	Mandal et al.	364/514
5633999	May 1997	Clowes et al.	395/182.04

FOREIGN PATENT DOCUMENTS

FOREIGN-PAT-NO	PUBN-DATE	COUNTRY	US-CL
633694	January 1995	EP	
WO93/16557	August 1993	WO	
WO95/10918	April 1995	WO	

OTHER PUBLICATIONS

Rangen PV, Vin HM, "Designing file systems for digital audio and video," Proceedings of the 13th ACM Symposium on Operating Systems Principles, Monterey, Calif., 1991, pp. 81-94.
Vin HM, Rangan PV, (1993), "Designing a multiuser HDTV storage service," IEEE Journal on Selected Areas in Communication, vol. 11, No. 1, Jan. 1993, pp. 153-164.

h e b b g e e e f e ef b e

Anderson DP, Osawa Y, Govindan R, "A file system for continuous media," ACM Transactions on Computer Systems, vol. 10., No. 4, Nov. 1992, pp. 311-337.
Federighi C, "A Distributed Hierarchical Storage Manager for a Video-on-Demand System," Department of Electrical Engr. and Computer Science, University of California, Berkeley, California, Dec. 1993.
Hastings R, The shark continuous-media file server. Proceedings IEEE COMPCON 93, San Francisco, Calif., 1993, pp. 12-15.
Little TD, Rhanger G, Folz RJ, Gibbon JF, Reeve FW, Schelleng DH, Venkatesh D, "A digital on-demand video service supporting content based queries," Proceedings of ACM Multimedia 93, Anaheim, Calif., Aug. 1-6, 1993, pp. 427-436.
Lougher, P, Sheperd, D. "The design of a storage server for continuous media," The Computer Journal, vol. 36, No. 1, 1993, pp. 32-42.
Rangan PV, Vin HM, Ramanathan S, "Designing an on-demand multimedia service," IEEE Communications Magazine, vol. 30, No. 7, Jul. 1992, pp. 56-64.
Sincoskie, WD, "System architecture for a large scale video on demand service," Computer Networks and ISDN Systems, vol. 22, No. 2, Sep. 1991, pp. 155-162.
Tobagi FA, Pang J, "StarWorks (Trademark)--A video applications server," Proceedings, IEEE COMPCON 93, San Francisco, Calif., 1993, pp. 4-11.
Vaitzblit L, "The design and implementation of a high bandwidth file service for continuous media," Master's Thesis, Massachusetts Institute of Technology, Cambridge, Mass., Nov. 4, 1991.

ART-UNIT: 272

PRIMARY-EXAMINER: Swann; Tod R.

ASSISTANT-EXAMINER: Tzeng; Fred F.

ATTY-AGENT-FIRM: Arnold, White & Durkee

ABSTRACT:

A video file server includes an integrated cached disk array storage subsystem and a plurality of stream server computers linking the cached disk array to a data network for the transfer of video data streams. The video file server further includes a controller server for applying an admission control policy to client requests and assigning stream servers to service the client requests. The stream servers include a real-time scheduler for scheduling isochronous tasks, and supports at least one industry standard network file access protocol and one file access protocol for continuous media file access. The cached disk storage subsystem is responsive to video prefetch commands, and the data specified for a prefetch command for a process are retained in an allocated portion of the cache memory from the time that the cached disk storage subsystem has responded to the prefetch command to the time that the cached disk storage subsystem responds to a fetch command specifying the data for the process. The time between prefetching and fetching is selected based on available disk and cache resources. The video file server provides video-on-demand service by maintaining and dynamically allocating sliding windows of video data in the random access memories of the stream server computers. The video file server has a tape silo for providing network backup services, and data to be written to tape are prestaged in the cached disk storage subsystem.

34 Claims, 20 Drawing figures

---

h e b b g e e e f e ef b e

☐ 10. Document ID: US 5805821 A

L11: Entry 10 of 16 File: USPT Sep 8, 1998

US-PAT-NO: 5805821
DOCUMENT-IDENTIFIER: US 5805821 A

TITLE: Video optimized media streamer user interface employing non-blocking switching to achieve isochronous data transfers

DATE-ISSUED: September 8, 1998

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Saxena; Ashok Raj	San Jose	CA		
Wang; Pong-Sheng	San Jose	CA		

ASSIGNEE-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY	TYPE CODE
International Business Machines Corporation	Armonk	NY			02

APPL-NO: 08/ 906567 [PALM]
DATE FILED: August 5, 1997

PARENT-CASE:
CROSS REFERENCE TO RELATED PATENT APPLICATION This is a continuation of application Ser. No. 08/303,190 filed on Sep. 8, 1994, now abandoned. This application is related to the following U.S. patent applications: Ser. No. 08/302,625, filed Sep. 8, 1994, entitled "Video Optimized Media Streamer", Inventors: W. R. Belknap et al.; Ser. No. 08/302,616, filed Sep. 8, 1994, entitled "Video Optimized Media Streamer for Generating Isochronous Data Streams", Inventors: W. R. Belknap et al.; Ser. No. 08/302,626, filed Sep. 8, 1994, entitled "Video Optimized Media Streamer Data Flow Architecture", Inventors: M. Henley et al.; Ser. No. 08/302,619, filed Sep. 8, 1994, entitled "Video Optimized Media Streamer with Cache Management", Inventors: W. R. Belknap et al.; and Ser. No. 08/302,624, filed Sep. 8, 1994, entitled "Video Optimized Media Streamer with Distributed Video Data Storage".

INT-CL: [06] H01 J 13/00

US-CL-ISSUED: 395/200.61; 395/826, 395/309, 395/311, 395/200.43, 395/200.62, 395/200.79
US-CL-CURRENT: 709/231; 709/213, 709/232, 709/249, 710/6

FIELD-OF-SEARCH: 395/200.38, 395/200.42, 395/200.43, 395/200.44, 395/200.46, 395/200.57, 395/200.59, 395/200.6, 395/200.61, 395/200.62, 364/940.92, 364/940.68

PRIOR-ART-DISCLOSED:

U.S. PATENT DOCUMENTS

PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL
4616263	October 1986	Eichelberger	358/12

h e b b g e e e f e ef b e

4679191	July 1987	Nelson et al.	370/84
4949187	August 1990	Cohen	358/335
5027400	June 1991	Baji et al.	
5088091	February 1992	Schroeder et al.	370/94.3
5089885	February 1992	Clark	358/86
5133079	July 1992	Ballantyne et al.	455/4.1
5200989	April 1993	Milone	379/53
5247347	September 1993	Litteral et al.	358/85
5283639	February 1994	Esch et al.	348/6
5289461	February 1994	De Nijs	370/58.1
5307490	April 1994	Davidson et al.	395/650
5311509	May 1994	Heddes et al.	370/60
5371532	December 1994	Gelman et al.	348/7
5428730	June 1995	Baker et al.	395/154
5442390	August 1995	Hooper et al.	348/7
5452419	September 1995	Di Giulio et al.	395/200.01
5473362	December 1995	Fitzgerald et al.	348/7
5491800	February 1996	Goldsmith et al.	395/200.12
5521631	May 1996	Budow et al.	348/7
5528513	June 1996	Vaitzblit et al.	364/514A
5550982	August 1996	Long et al.	395/200.13
5586257	December 1996	Perlman	432/42
5606359	February 1997	Youden et al.	348/7
5668948	September 1997	Belknap et al.	395/200.61

FOREIGN PATENT DOCUMENTS

FOREIGN-PAT-NO	PUBN-DATE	COUNTRY	US-CL
2117422	August 1998	CA	
529864A1	March 1993	EP	
0 529 864	March 1993	EP	
0 535 807	April 1993	EP	
WO93/16557	August 1993	WO	
WO 94/01964	January 1994	WO	
2071416	January 1994	WO	

OTHER PUBLICATIONS

European Search Report, Application No. EP 95 30 5967, dated Dec. 29, 1995.
Interactive Video On Demand, Article by Daniel Deloddere et al, pp. 82-88 as appeared in IEEE Communications Magazine, May 1994.
Star Works--A Video Applications Server, Article by Tobagi et al. of Starlight Networks, Inc. as appeared in IEEE Feb., 1993, pp. 4-11.
Allicat (0664) S10 SCSI Direct Access Storage Device Functional Specification Release 4.50, Docment No. AS01-0003-00, Formatted on May 6, 1993 IBM Corporation, pp. 1-291.
"Intel Scalable Multi-server Technology for Interactive Multimedia Applications" May, 1994, 1994 Intel Corporation, pp. 3-15.
"Architectures for Video Servers" by Manu Thapar and Bill Koerner, Hewlett Packard,

1994 NCTA Technical Papers, pp. 141-148.

ART-UNIT: 271

PRIMARY-EXAMINER: Sheikh; Ayaz R.

ASSISTANT-EXAMINER: Phan; Raymond N.

ATTY-AGENT-FIRM: Ohlandt, Greeley, Ruggiero & Perle

ABSTRACT:

A media streamer (10) includes at least one control node (18); a user interface (64, 65, 66) having an output coupled to the at least one control node; at least one storage node (16, 17) for storing a digital representation of at least one video presentation; and a plurality of communication nodes (14) each having an input port for receiving a digital representation of at least one video presentation therefrom. The video presentation requires a time T to present in its entirety, and is stored as a plurality of N data blocks. Each data block stores data corresponding to a T/N period of the video presentation. Each communication nodes further has a plurality of output ports for outputting a digital representation. A circuit switch (12) is connected between the at least one storage node and the input ports of communication nodes for coupling one or more input ports to the at least one storage node. The user interface includes a capability for specifying commands for execution, and the at least one control node is responsive to individual ones of the commands for controlling at least one of the at least one storage node and at least one of the plurality of communication nodes, in cooperation with the circuit switch, so as to execute a function associated with individual ones of the commands. The commands may include video cassette recorder-like commands that include commands selected from a group that includes a Load command, an Eject command, a Play command, a Slow command, a Fast Forward command, a Pause command, a Stop command, a Rewind command, and a Mute command. The commands may also include commands selected from a group that includes a Play List command, a Play Length command, and a Batch command. A synchronous application program interface (API) is provided for coupling, via the user interface, a user application program to the at least one control node. The API includes Remote Procedure Call (RPC) procedures.

7 Claims, 26 Drawing figures

☐ 11. Document ID: US 5761417 A

L11: Entry 11 of 16 File: USPT Jun 2, 1998

US-PAT-NO: 5761417
DOCUMENT-IDENTIFIER: US 5761417 A

TITLE: Video data streamer having scheduler for scheduling read request for individual data buffers associated with output ports of communication node to one storage node

DATE-ISSUED: June 2, 1998

INVENTOR-INFORMATION:

h e b b g e e e f e ef b e

NAME	CITY	STATE	ZIP CODE	COUNTRY
Henley; Martha R.	Morgan Hill	CA		
Wyllie; James Christopher	Monte Sereno	CA		
Saxena; Ashok Raj	San Jose	CA		

ASSIGNEE-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY	TYPE CODE
International Business Machines Corporation.	Armonk	NY			02

APPL-NO: 08/ 302626 [PALM]
DATE FILED: September 8, 1994

INT-CL: [06] G06 F 13/00

US-CL-ISSUED: 395/200.09; 348/7, 455/4.2
US-CL-CURRENT: 709/231; 725/117, 725/92, 725/94, 725/97

FIELD-OF-SEARCH: 395/250, 395/550, 395/200.01, 395/200.02, 395/200.03, 395/200.06, 395/200.08, 395/200.09, 395/405, 395/441, 395/454, 358/310, 358/312, 358/322, 358/340, 358/341, 358/342, 358/146, 360/4, 360/13, 360/14.1, 360/18, 369/13, 369/14, 369/15, 369/16, 369/17, 369/18, 369/176, 369/178, 348/7, 455/3.1, 455/4.1, 455/4.2, 455/5.1

PRIOR-ART-DISCLOSED:

U.S. PATENT DOCUMENTS

PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL
4355324	October 1982	Reitmeier	386/6
4616263	October 1986	Eichelberger	348/722
4679191	July 1987	Nelson et al.	370/355
4949187	August 1990	Cohen	358/69
5089885	February 1992	Clark	348/7
5133079	July 1992	Ballantyne et al.	455/4.1
5200989	April 1993	Milone	379/96
5283639	February 1994	Esch et al.	348/6
5289461	February 1994	de Nijs	370/360
5421031	May 1995	De Bey	455/5.1
5461415	October 1995	Wolf et al.	348/7
5471640	November 1995	McBride	395/842
5473362	December 1995	Fitzgerald et al.	348/7
5487167	January 1996	Dinallo et al.	395/807
5510905	April 1996	Birk	386/125

FOREIGN PATENT DOCUMENTS

FOREIGN-PAT-NO	PUBN-DATE	COUNTRY	US-CL
0529-864-A1	March 1993	EP	

h e b b g e e e f e ef b e

WO93/16557 August 1993 WO

OTHER PUBLICATIONS

Gelman et al, "On Buffer Requirements for Store-and-Forward Video on Demand", IEEE, pp. 0976-0980, 1991.
Papadimitriou et al, "Information Caching for Delivery of Personalized Video Programs on Home Entertainment Channels", IEEE, pp. 214-223, 1994.
Gelman et al, "A Store-and-Forward Architecture for Video-on-Demand Service", IEEE, pp. 0842-0846, 1991.
Hosekote, "Scheduling Continous Media in a Video-on-Demand Server", IEEE, pp. 19-28, 1994.
Allicat (0664) S10 SCSI Direct Access Storage Device Functional Specification Release 4.50, Document No. AS01-0003-00, Formatted on May 6, 1993. IBM Corporation, pp. 1-291.
"Intel Scalable Multi-server Technology for Interactive Multimedia Applications" May, 1994, 1994 Intel Corporation, pp. 3-15.
"Architectures for Video Servers" by Manu Thapar and Bill Koerner, Hewlett Packard, 1994 NCTA Technical Papers, pp. 141-148.

ART-UNIT: 237

PRIMARY-EXAMINER: Meky; Moustafa M.

ATTY-AGENT-FIRM: Ohlandt, Greeley, Ruggiero & Perle

ABSTRACT:

A media streamer (10) includes at least one storage node (16, 17) comprising mass storage for storing a digital representation of at least one video presentation; and a plurality of communication nodes (14) each having at least one input port that is coupled to an output of the at least one storage node for receiving a digital representation of a video presentation therefrom. Each of the plurality of communication nodes further includes a plurality of output ports, individual ones of the plurality of output ports being operable for transmitting a digital representation as a data stream to a consumer of the digital representation. Individual ones of the output ports also have an associated data buffer for buffering a portion of a digital representation prior to a transmission of the digital representation. Each of the plurality of communication nodes further includes a scheduler for scheduling, for individual ones of the data buffers, a read request to the at least one storage node for a next sequential portion of the digital representation for storage within individual ones of the data buffers. Each read request for a particular individual one of the data buffers is scheduled at a predetermined time such that a requested next sequential portion of the digital representation is available to a requesting communications node prior to a time that the requested next sequential portion is required to be transmitted from the output port that is associated with the data buffer for which the read request was scheduled. The at least one storage node further includes a scheduler for scheduling a read request to the mass storage such that a next sequential portion of a digital representation is available for outputting, prior to the at least one storage node receiving a read request for the next sequential portion from one of the communication nodes.

7 Claims, 26 Drawing figures

h e b b g e e e f e ef b e

☐ 12. Document ID: US 5737747 A

L11: Entry 12 of 16 File: USPT Apr 7, 1998

US-PAT-NO: 5737747
DOCUMENT-IDENTIFIER: US 5737747 A

TITLE: Prefetching to service multiple video streams from an integrated cached disk array

DATE-ISSUED: April 7, 1998

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Vishlitzky; Natan	Brookline	MA		
Wilson; Robert	Hopkinton	MA		
Tzelnic; Percy	Concord	MA		

ASSIGNEE-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY	TYPE CODE
EMC Corporation	Hopkinton	MA			02

APPL-NO: 08/ 661185 [PALM]
DATE FILED: June 10, 1996

INT-CL: [06] G06 F 13/16

US-CL-ISSUED: 711/118; 711/113, 364/DIG.1
US-CL-CURRENT: 711/118; 711/113

FIELD-OF-SEARCH: 395/200.08, 395/200.09, 395/800, 395/855, 395/445, 364/514A, 348/12

PRIOR-ART-DISCLOSED:

U.S. PATENT DOCUMENTS

PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL
5208665	May 1993	McCalley et al.	348/12
5218695	June 1993	Noveck et al.	395/600
5276860	January 1994	Fortier et al.	395/575
5276867	January 1994	Kenley et al.	395/600
5367698	November 1994	Webber et al.	395/800
5371532	December 1994	Gelman et al.	348/7
5442771	August 1995	Filepp et al.	395/200.09
5528513	June 1996	Vailebut et al.	364/514A
5544313	August 1996	Shachnai	395/200.1
5583561	December 1996	Baker et al.	348/7
5586264	December 1996	Belknap et al.	395/200.08
5594910	January 1997	Filepp et al.	395/800

h e b b g ee e f e ef b e

5603058 February 1997 Belknap et al. 395/855

FOREIGN PATENT DOCUMENTS

FOREIGN-PAT-NO	PUBN-DATE	COUNTRY	US-CL
633694	January 1995	EP	
W093/16557	August 1993	WO	
W095/10918	April 1995	WO	

OTHER PUBLICATIONS

Rangen PV, Vin HM, "Designing file systems for digital audio and video," Proceedings of the 13th ACM Symposium on Operating Systems Principles, Monterey, Calif., 1991, pp. 81-94.
Vin HM, Rangan PV, (1993), "Designing a multiuser HDTV storage service," IEEE Journal on Selected Areas in Communication, vol. 11, No. 1, Jan. 1993, pp. 153-164.

Anderson DP, Osawa Y, Govindan R, "A file system for continuous media," ACM Transactions on Computer Systems, vol. 10., No. 4, Nov. 1992, pp. 311-337.
Federighi C, "A Distributed Hierarchical Storage Manager for a Video-on-Demand System," Department of Electrical Engr. and Computer Science, University of California, Berkeley, California, Dec. 1993.
Hastings R, The shark continuous-media file server. Proceedings IEEE COMPCON 93, San Francisco, Calif., 1993, pp. 12-15.
Little TD, Rhanger G, folz RJ, Gibbon JF, Reeve FW, Schelleng DH, Venkatesh D, "A digital on-demand video service supporting content based queries," Proceedings of ACM Multimedia 93, Anaheim, Calif., Aug. 1-6, 1993, pp. 427-436.
Lougher, P, Shepard, D. "The design of a storage server for continuous media," The Computer Journal, vol. 36, No. 1, 1993, pp. 32-42.
Rangan PV, Vin HM, Ramanathan S, "Designing an on-demand multimedia service," IEEE Communications magazine, Vol. 30, No. 7, Jul. 1992, pp. 56-64.
Sincoskie, WD, "System architecture for a large scale video on demand service," Computer Networks and ISDN Systems, Vol. 22, No. 2, Sep. 1991, pp. 155-162.
Tobagi FA, Pang J, "Starworks (Trademark) --A video applications server," Proceedings, IEEE COMPCON 93, San Francisco, Calif., 1993, pp. 4-11.
Vaitzblit L, "The design and implementation of a high band-width file service for continuous media, " Master's Thesis, Massachusetts Institute of Technology, Cambridge, Mass., Nov. 4, 1991.

ART-UNIT: 238

PRIMARY-EXAMINER: Swann; Tod R.

ASSISTANT-EXAMINER: Langjahr; David C.

ATTY-AGENT-FIRM: Arnold, White & Durkee

ABSTRACT:

A video file server includes an integrated cached disk array storage subsystem and a plurality of stream server computers linking the cached disk storage subsystem to the data network for the transfer of video data streams. The video file server further includes a controller server for applying an admission control policy to client requests and assigning stream servers to service the client requests. The stream servers include a real-time scheduler for scheduling isochronous tasks, and supports at least one industry standard network file access protocol and one file

access protocol for continuous media file access. The cached disk storage subsystem is responsive to video prefetch commands, and the data specified for a prefetch command for a process are retained in an allocated portion of the cache memory from the time that the cached disk storage subsystem has responded to the prefetch command to the time that the cached disk storage subsystem responds to a fetch command specifying the data for the process. The time between prefetching and fetching is selected based on available disk and cache resources. The video file server provides video-on-demand service by maintaining and dynamically allocating sliding windows of video data in the random access memories of the stream server computers.

35 Claims, 20 Drawing figures

☐ 13. Document ID: US 5712976 A

L11: Entry 13 of 16 File: USPT Jan 27, 1998

US-PAT-NO: 5712976
DOCUMENT-IDENTIFIER: US 5712976 A

TITLE: Video data streamer for simultaneously conveying same one or different ones of data blocks stored in storage node to each of plurality of communication nodes

DATE-ISSUED: January 27, 1998

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Falcon, Jr.; Lorenzo	San Jose	CA		
Saxena; Ashok Raj	San Jose	CA		

ASSIGNEE-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY	TYPE CODE
International Business Machines Corporation	Armonk	NY			02

APPL-NO: 08/ 302624 [PALM]
DATE FILED: September 8, 1994

PARENT-CASE:
CROSS REFERENCE TO RELATED PATENT APPLICATIONS This application is related to the following U.S. patent applications: Ser. No. 08/302,625, filed Sep. 8, 1994, still pending, entitled "Video Optimized Media Streamer", Inventors: W. R. Belknap et al.; Ser. No. 08/302,616, filed Sep. 8, 1994, now U.S. Pat. No. 5,603,058, entitled "Video Optimized Media Streamer for Generating Isochronous Data Streams", Inventors: W. R. Belknap et al.; Ser. No. 08/302,626, filed Sep. 8, 1994, still pending, entitled "Video Optimized Media Streamer Data Flow Architecture", Inventors: M. Henley et al.; Ser. No. 08/302,619, filed Sep. 8, 1994, now U.S. Pat. No. 5,586,264, entitled "Video Optimized Media Streamer with Cache Management", Inventors: W. R. Belknap et al.; and Ser. No. 08/303,190, filed Sep. 8 1994, still pending, entitled "Video Optimized Media Streamer User Interface", Inventors: A. Saxena et al.

INT-CL: [06] G06 F 13/00

US-CL-ISSUED: 395/200.09; 348/7, 455/4.2
US-CL-CURRENT: 725/115; 709/231, 725/114

FIELD-OF-SEARCH: 395/250, 395/550, 395/650, 395/154, 395/441, 395/200.01, 395/200.02, 395/200.03, 395/200.06, 395/200.08, 395/200.09, 395/405, 395/454, 358/310, 358/312, 358/322, 358/340, 358/341, 358/342, 358/146, 369/13, 369/14, 369/15, 369/16, 369/17, 369/18, 369/176, 369/178, 348/7, 455/3.1, 455/4.1, 455/4.2, 455/5.1

PRIOR-ART-DISCLOSED:

U.S. PATENT DOCUMENTS

PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL
4355324	October 1982	Reitmeier	386/6
4604687	August 1986	Abbott	395/616
4616263	October 1986	Eichelberger	348/722
4679191	July 1987	Nelson et al.	370/355
4949187	August 1990	Cohen	386/69
5089885	February 1992	Clark	348/7
5133079	July 1992	Ballantyne et al.	455/4.1
5200989	April 1993	Milone	348/16
5283639	February 1994	Esch et al.	348/6
5289461	February 1994	de Nijs	370/360
5421031	May 1995	De Bey	455/5.1
5442390	August 1995	Hooper et al.	348/7
5461415	October 1995	Wolf et al.	348/7
5471640	November 1995	McBride	395/842
5473362	December 1995	Fitzgerald et al.	348/7
5487167	January 1996	Dinallo et al.	395/807
5510905	April 1996	Birk	386/125
5513011	April 1996	Matsumoto et al.	386/98
5522054	May 1996	Gunlock et al.	395/439
5526507	June 1996	Hill	395/441
5528513	June 1996	Vaitzblit et al.	364/514A
5530830	June 1996	Iwasaki et al.	395/441
5544327	August 1996	Dan et al.	395/250
5550982	August 1996	Long et al.	395/200.13
5586264	December 1996	Belknap et al.	395/200.08
5606359	February 1997	Youden et al.	348/7

FOREIGN PATENT DOCUMENTS

FOREIGN-PAT-NO	PUBN-DATE	COUNTRY	US-CL
0529-864-A1	March 1993	EP	
WO93/16557	August 1993	WO	

h e b b g e e e f e ef b e

OTHER PUBLICATIONS

"Intel Scalable Multi-server Technology for Interactive Multimedia Applications" May, 1994, 1994 Intel Corporation, pp. 3-15.
"Architectures for Video Servers" by Manu Thapar and Bill Koerner, Hewlett Packard, 1994 NCTA Technical Papers, pp. 141-148.
ALLICAT (0664) S10 SCSI Direct Access Storage Device Functional Specification Release 4.50, Document No. AS01-0003-00, Formatted on May 6, 1993 IBM Corporation, pp. 1-291.
Hosekote et al, "Scheduling continous media in a video-on-demand server", IEEE, pp. 19-28, 1994.
Kim, Michelle, "Synchronized Disk Interleaving", IEEE, pp.978-988, Nov. 1986.
Rangan et al, "Designing File Systems for Digital Video and Audio", ACM Press, pp.91-104, Oct. 13, 1991.

ART-UNIT: 237

PRIMARY-EXAMINER: Meky; Moustafa M.

ATTY-AGENT-FIRM: Ohlandt, Greeley, Ruggiero & Perle

ABSTRACT:

A media streamer (10) includes at least one storage node (16) including mass storage for storing a digital representation of at least one video presentation. The mass storage is comprised of a plurality of mass storage units. The at least one video presentation requires a time T to present in its entirety, and is stored as a plurality of N data blocks. Each data block stores data corresponding to approximately a T/N period of the video presentation. The media streamer further includes a plurality of communication nodes (14) each having at least one input port that is coupled to an output of the at least one storage node for receiving a digital representation of a video presentation therefrom. Each of the plurality of communication nodes further includes a plurality of output ports individual ones of which transmit a digital representation as a data stream to a consumer of the digital representation. The N data blocks of the digital representation are partitioned into X stripes, wherein data blocks 1, X+1, 2*X+1, . . . etc., are associated with a first one of the X stripes, data blocks 2, X+2, 2*X+2, . . . etc., are associated with a second one of the X stripes, etc., and wherein individual ones of the X stripes are each stored on a different one of the plurality of mass storage units. The plurality of mass storage units preferably store a single copy of a digital representation of a video presentation. The X stripes are read out in such a manner as to enable a plurality of data streams to simultaneously convey a same one of the N data blocks, or are read out in such a manner as to enable a plurality of data streams to simultaneously convey a different one of the N data blocks.

4 Claims, 26 Drawing figures

☐ 14. Document ID: US 5668948 A

L11: Entry 14 of 16 File: USPT Sep 16, 1997

US-PAT-NO: 5668948
DOCUMENT-IDENTIFIER: US 5668948 A

h e b b g e e e f e ef b e

TITLE: Media streamer with control node enabling same isochronous streams to appear simultaneously at output ports or different streams to appear simultaneously at output ports

DATE-ISSUED: September 16, 1997

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Belknap; William Russell	San Jose	CA		
Cleary; Louise Irene	Boca Raton	FL		
Eldridge; James W.	San Jose	CA		
Fitchett; Larry William	Morgan Hill	CA		
Luning; Stephen G.	San Jose	CA		
Murray; Christopher S.	Boynton Beach	FL		
Olnowich; Howard T.	Endwell	NY		
Saxena; Ashok Raj	San Jose	CA		
Schubert; Karl David	San Jose	CA		
Stansbury; Buddy Floyd	San Jose	CA		

ASSIGNEE-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY	TYPE CODE
International Business Machines Corporation	Armonk	NY			02

APPL-NO: 08/ 302625 [PALM]
DATE FILED: September 8, 1994

INT-CL: [06] G06 F 9/00

US-CL-ISSUED: 395/200.61; 395/826, 395/672, 395/684, 395/311, 395/200.79, 348/552
US-CL-CURRENT: 709/231; 348/552, 709/249, 710/316, 710/6, 718/102

FIELD-OF-SEARCH: 395/200.12, 364/242.5, 364/243.6, 364/919.2, 364/919.5, 364/935.4, 364/940.92, 364/940.64, 364/940.68, 398/7

PRIOR-ART-DISCLOSED:

U.S. PATENT DOCUMENTS

PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL
4679191	July 1987	Nelson et al.	370/355
4949187	August 1990	Cohen	386/69
5027400	June 1991	Baji et al.	380/20
5166926	November 1992	Cisneros et al.	370/392
5200989	April 1993	Milone	348/16
5247347	September 1993	Litteral et al.	358/85
5283639	February 1994	Esch et al.	348/6
5289461	February 1994	de Nijs	370/360
5357276	October 1994	Banker et al.	348/7
5404537	April 1995	Olnowich et al.	395/725

h e b b g e e e f e ef b e

5414455	May 1995	Hooper et al.	348/7
5421031	May 1995	De Bey	455/5.1
5442390	August 1995	Hooper et al.	348/7
5453779	September 1995	Dan et al.	348/7
5461415	October 1995	Wolf et al.	348/7
5473362	December 1995	Fitzgerald	348/7
5487035	January 1996	Nishimura et al.	365/189.02
5528513	June 1996	Vaitzblit et al.	364/514A

FOREIGN PATENT DOCUMENTS

FOREIGN-PAT-NO	PUBN-DATE	COUNTRY	US-CL
2117422	August 1993	CA	
0 368 683	May 1990	EP	
0529-864-A1	March 1993	EP	
0 535 807	April 1993	EP	
0 605 115	July 1994	EP	
0617563	September 1994	EP	
WO9103112	March 1991	WO	
WO94/01964	June 1993	WO	
WO93/16557	August 1993	WO	

OTHER PUBLICATIONS

"Intel Scalable Multi-server Technology for Interactive Multimedia Applications" May, 1994, 1994 Intel Corporation, pp. 3-15.
"Architectures for Video Servers" by Manu Thapar and Bill Koerner, Hewlett Packard, 1994 NCTA Technical Papers, pp. 141-148.
Allicat (0664) S10 SCSI Direct Access Storage Device Functional Specification Release 4.50, Document No. AS01-0003-00, Formatted on May 6, 1993 IBM Corporation, pp. 1-291.

ART-UNIT: 237

PRIMARY-EXAMINER: Lee; Thomas C.

ASSISTANT-EXAMINER: Chen; Anderson I.

ATTY-AGENT-FIRM: Ohlandt, Greeley, Ruggiero & Perle

ABSTRACT:

A media streamer (10) includes at least one storage node (16, 17) for storing a digital representation of a video presentation. The Video presentation requires a time T to present in its entirety, and is stored as a plurality of N data blocks, each data block storing data corresponding approximately to a T/N period of the video presentation. The media streamer further includes a plurality of communication nodes (14) each having at least one input port and at least one output port; a circuit switch (18) connected between the at least one storage node and input ports of the plurality of communication nodes, the circuit switch selectively coupling one or more of the input ports to the at least one storage node to enable the digital representation stored thereat to appear at one or more

h e b b g e e e f e ef b e

of the output ports; and at least one control node (18) coupled at least to the plurality of communication nodes and to the at least one storage node for enabling any one of the N blocks to appear at any output port of any of the plurality of communication nodes.

14 Claims, 26 Drawing figures

☐ 15. Document ID: US 5603058 A

L11: Entry 15 of 16 File: USPT Feb 11, 1997

US-PAT-NO: 5603058
DOCUMENT-IDENTIFIER: US 5603058 A

TITLE: Video optimized media streamer having communication nodes received digital data from storage node and transmitted said data to adapters for generating isochronous digital data streams

DATE-ISSUED: February 11, 1997

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Belknap; William R.	San Jose	CA		
Fitchett; Larry W.	Morgan Hill	CA		
Stansbury; Buddy F.	San Jose	CA		

ASSIGNEE-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY	TYPE CODE
International Business Machines Corporation	Armonk	NY			02

APPL-NO: 08/ 302616 [PALM]
DATE FILED: September 8, 1994

PARENT-CASE:
CROSS REFERENCE TO RELATED PATENT APPLICATIONS This application is related to the following U.S. patent applications. Ser. No. 08/302,625, filed Sep. 8, 1994, entitled "Video Optimized Media Streamer", Inventors: W. R. Belknap et al.; Ser. No. 08/302,626, filed Sep. 8, 1994, entitled "Video Optimized Media Streamer Data Flow Architecture", Inventors: M. Henley et al.; Ser. No. 08/302,619, filed Sep. 8, 1994, entitled "Video Optimized Media Streamer with Cache Management", Inventors: W. R. Belknap et al.;

INT-CL: [06] G06 F 13/00

US-CL-ISSUED: 395/855; 395/200.08, 395/858, 360/39
US-CL-CURRENT: 710/35; 360/39, 709/231, 710/38

FIELD-OF-SEARCH: 360/19.1, 360/8, 360/15, 380/10, 348/12, 371/37.1, 358/143

PRIOR-ART-DISCLOSED:

h e b b g e e e f e ef b e

U.S. PATENT DOCUMENTS

PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL
4616263	October 1986	Eichelberger	358/12
4679191	July 1987	Nelson et al.	370/84
4772959	September 1988	Amano et al.	360/8
4851931	July 1989	Parker et al.	360/15
4949187	August 1990	Cohen	358/335
5014310	May 1991	Walker et al.	380/10
5027400	June 1991	Baji et al.	380/20
5089885	February 1992	Clark	358/86
5133079	July 1992	Ballantyne et al.	455/4.1
5200989	April 1993	Milone	379/53
5231492	July 1993	Dangi et al.	358/143
5239540	August 1993	Rovira et al.	370/77
5253120	October 1993	Endoh	360/19.1
5283639	February 1994	Esch et al.	348/6
5289461	February 1994	de Nijs	370/58.1
5374952	December 1994	Flohr	348/12
5432799	July 1995	Shimpuku et al.	371/37.1
5485197	January 1996	Hoarty	395/159
5517652	May 1996	Miyamoto et al.	395/800

FOREIGN PATENT DOCUMENTS

FOREIGN-PAT-NO	PUBN-DATE	COUNTRY	US-CL
2117422	August 1993	CA	
2071416	December 1993	CA	
0368683	May 1990	EP	
0529864A1	March 1993	EP	
WO93/16557	August 1993	WO	
WO94/12937	June 1994	WO	

OTHER PUBLICATIONS

"Intel Scalable Multi-server Technology for Interactive Multimedia Applications" May, 1994, 1994 Intel Corporation, pp. 3-15.
"Architectures for Video Servers" by Manu Thapar and Bill Koerner, Hewlett Packard, 1994 NCTA Technical Papers, pp. 141-148.
Allicat (0664) S10 SCSI Direct Access Storage Device Functional Specification Release 4.50, Document No. AS01-0003-00, Formatted on May 6, 1993 IBM Corporation, pp. 1-291.
Article entitled Interactive Video on Demand by Daniel Deloddere et al. as appeared in IEEE Communications Magazine, May 1994, pp. 82-88.
Article entitled Scheduling Real-Time Disk Transfers for Continuous Media Applications by Darrell D. E. Long et al., pp. 227-232, Proceedings of 12th IEEE Symposium on Mass Storage Systems--Apr. 26-29, 1993 in Monterey, CA, USA.
Article entitled Scheduling Continuous Media in a Video-on-Demand Server by Kenchammana-Hosekote et al., pp. 19-28, Proceedings of IEEE International

h e b b g e e e f e ef b e

Conference on Multimedia Computing and Systems, Boston, MA, USA, 15-19 May 1994.

ART-UNIT: 237

PRIMARY-EXAMINER: Lee; Thomas C.

ASSISTANT-EXAMINER: Luu; Le Hien

ATTY-AGENT-FIRM: Ohlandt, Greeley, Ruggiero & Perle

ABSTRACT:

A media streamer (10) includes at least one control node (18); at least one storage node (16, 17) for storing a digital representation of a video presentation; and a plurality of communication nodes (14) each having an input port that is switchably coupled under the direction of the control node to an output of the at least one storage node for receiving a digital representation of a video presentation therefrom. Each of the plurality of communication nodes further includes at least one output port for coupling to a first end of a communications bus (210). Individual ones of the communication nodes output a digital representation of a video presentation as a sequence of data bursts to the first end of the communications bus. The media streamer further includes an adapter (15), coupled to a second end of the communications bus, for receiving the sequence of data bursts and for converting the received sequence of data bursts to a substantially isochronous data stream that represents a video presentation.

13 Claims, 25 Drawing figures

☐ 16. Document ID: US 5586264 A

L11: Entry 16 of 16 File: USPT Dec 17, 1996

US-PAT-NO: 5586264
DOCUMENT-IDENTIFIER: US 5586264 A

TITLE: Video optimized media streamer with cache management

DATE-ISSUED: December 17, 1996

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Belknap; William R.	San Jose	CA		
Henley; Martha R.	Morgan Hill	CA		
Falcon, Jr.; Lorenzo	San Jose	CA		
Frayne; Thomas E.	San Jose	CA		
Luo; Mei-Lan	San Jose	CA		
Saxena; Ashok R.	San Jose	CA		

ASSIGNEE-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY	TYPE CODE
IBM Corporation	San Jose	CA			02

h e b b g e e e f e ef b e

APPL-NO: 08/ 302619 [PALM]
DATE FILED: September 8, 1994

INT-CL: [06] H04 N 7/14, G06 F 15/00

US-CL-ISSUED: 395/200.08; 395/800, 395/872, 395/834, 348/7, 364/DIG.1, 364/DIG.2, 364/239, 364/238.3, 364/241.9, 364/222.2, 364/939, 364/260, 364/242.94
US-CL-CURRENT: 725/115; 709/231, 710/14, 710/52, 711/118, 725/119, 725/88, 725/92, 725/94, 725/97

FIELD-OF-SEARCH: 348/7, 370/60, 370/94.1, 364/DIG.1, 364/DIG.2, 364/239, 364/234, 364/236.3, 364/236.2, 364/248.1, 364/260, 364/238.4, 364/254, 364/259, 364/241.9, 364/271, 364/939, 395/600, 395/164, 395/439, 395/427, 395/441, 395/275, 395/100, 395/118, 395/154

PRIOR-ART-DISCLOSED:

U.S. PATENT DOCUMENTS

PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL
4616263	October 1986	Eichelberger	358/12
4679191	July 1987	Nelson et al.	370/84
4949187	August 1990	Cohen	358/335
5089885	February 1992	Clark	358/86
5133079	July 1992	Ballantyne et al.	455/4.1
5166930	November 1992	Braff et al.	370/94.1
5200989	April 1993	Milone	379/53
5283639	February 1994	Esch et al.	348/6
5289461	February 1994	de Nijs	370/58.1
5406556	April 1995	Widjaja et al.	370/60
5414455	May 1995	Hooper et al.	348/7
5442390	August 1995	Hooper et al.	348/7

FOREIGN PATENT DOCUMENTS

FOREIGN-PAT-NO	PUBN-DATE	COUNTRY	US-CL
2117422	August 1993	CA	
2071416	December 1993	CA	
0368683	May 1990	EP	
0529864A1	March 1993	EP	
2270791	August 1993	GB	
WO93/16557	August 1993	WO	
WO94/12937	June 1994	WO	

OTHER PUBLICATIONS

European Search Report, Application No. EP 95 30 5966, dated 29 Dec. 1995.
Article entitled Interactive Video On Demand by Daniel Deloddere et al. as appeared in IEEE Communications Magazine, May 1994, pp. 82-88.

h e b b g e e e f e ef b e

"Intel Scalable Multi-server Technology for Interactive Multimedia Applications" May, 1994, 1994 Intel Corporation, pp. 3-15.
"Architectures for Video Servers" by Manu Thapar and Bill Koerner, Hewlett Packard, 1994 NCTA Technical Papers, pp. 141-148.
Allicat (0664) S10 SCSI Direct Access Storage Device Functional Specification Release 4.50, Document No. AS01-0003-00, formatted on May 6, 1993 IBM Corporation, pp. 1-291.

ART-UNIT: 237

PRIMARY-EXAMINER: Lee; Thomas C.

ASSISTANT-EXAMINER: Stanton; Terance J.

ATTY-AGENT-FIRM: Ohlandt, Greeley, Ruggiero & Perle

ABSTRACT:

A data storage system includes a mass storage unit storing a data entity, such as a digital representation of a video presentation, that is partitioned into a plurality N of temporally-ordered segments. A data buffer is bidirectionally coupled to the mass storage unit for storing up to M of the temporally-ordered segments, wherein M is less than N. The data buffer has an output for outputting stored ones of the temporally-ordered segments. The data storage system further includes a data buffer manager for scheduling transfers of individual ones of the temporally-ordered segments between the mass storage unit and the data buffer. The data buffer manager schedules the transfers in accordance with at least a predicted time that an individual one of the temporally-ordered segments will be required to be output from the data buffer. When employed with a media streamer (10) distributed data buffer management techniques are employed for selecting blocks to be retained in a buffer memory, either in a storage node (16, 17) or in a communication node (14). These techniques rely on the predictable nature of the video data stream, and thus are enabled to predict the future requirements for a given one of the data blocks.

12 Claims, 26 Drawing figures

Term	Documents
(9 AND 10).USPT.	16
(L9 AND L10).USPT.	16

h e b b g e e e f e ef b e

First Hit Fwd Refs

Generate Collection Print

L8: Entry 5 of 9 File: USPT Oct 26, 1999

DOCUMENT-IDENTIFIER: US 5974503 A
TITLE: Storage and access of continuous media files indexed as lists of raid stripe sets associated with file names

Abstract Text (1):
A continuous media file is comprised of stripe sets over disk drives in one or more RAID sets. In a preferred embodiment, the RAID set includes n disk drives. The data storage of each disk drive in the RAID set is partitioned into an integer number m of hyper-volumes, and the parity is stored in one hyper-volume of each of m disk drives in the RAID set. The stripe set includes a series of transfer units of data in respective ones of the disk drives. Each transfer unit includes an integer number j of data blocks, and each hyper-volume includes an integer number k of transfer units. Each stripe set includes (m)(n-1) transfer units of data. The transfer units of the RAID set are allocated for the storage of continuous media data in a right-to-left and then top-to-bottom order in which the transfer units appear in an m row by n column matrix in which the rows of the matrix represent parity groups of hyper-volumes in the disk drives and the columns of the matrix represent storage in the respective disk drives. At most one write access to each parity hyper-volume need be performed during write access to a stripe set. Parity changes for the data being written are accumulated in non-volatile memory, and written to the RAID set after completion of the writing of the data.

Parent Case Text (2):
The present application is a continuation of provisional application Serial No. 60/044,948 filed Apr. 25, 1997 by Dinesh Venkatesh, Wayne W. Duso, John Forecast, Uday Gupta, Uresh K. Vahalia, and Dennis P. J. Ting, entitled "Raid Striping, Client-Server Protocols, and Failover Services for a Video File Server."

Brief Summary Text (5):
The present invention relates generally to a data storage system employing a redundant array of disk drives (RAID), and more particularly to a continuous media or video file server employing RAID techniques.

Brief Summary Text (7):
In a data storage system employing RAID techniques, a set of disk drives are configured into a RAID set storing data and parity computed across the disk drives in the RAID set. Therefore, if one of the disk drives in the RAID set fails, data in the failed disk drive can be reconstructed from the data and parity information in the other disk drives in the RAID set.

Brief Summary Text (8):
In a file server employing RAID techniques, it is conventional to employ a file directory for mapping a file name to a list of data blocks stored in respective ones of the disk drives in the RAID set. The data blocks are typically of a fixed size, such as 512 bytes, that are conveniently handled by the buffers and control circuitry provided by commodity 5 and 1/4 inch or 3 and 1/2 inch disk drives. A write access to a data block involves reading old data from an addressed disk drive, reading old parity corresponding to the old data from another disk drive in the raid set, computing a change in parity by an exclusive-OR of the old data with the new data to be written, computing the new parity by an exclusive-OR of the

change in parity with the old parity, and then, in a single transaction, writing the new data and the new parity back to the respective disk drives.

Brief Summary Text (9):
It is desirable to use a data storage system employing RAID techniques for storing continuous media such as digital video. This could be done by using the conventional RAID techniques described above for a file server. However, this has been found to cause computational inefficiencies and load balancing and data availability problems under normal operation and under a failure mode of operation when there has been a failure of one of the disk drives in the RAID set.

Brief Summary Text (11):
The computational inefficiencies, data availability problems, and load balancing problems associated with storing continuous media in a conventional RAID storage system result from the relatively large size, relatively high access rate, and non-uniform access frequency of continuous media files. Because continuous media files are typically accessed in a sequential or isochronous fashion, it is desirable to access a relatively large transfer unit of continuous media data during each disk access due to the fixed overhead of managing each disk access. However, it is very likely that one continuous media file, such as a file for a popular movie, could exceed the capacity of a single disk drive, and could have an access frequency exceeding the combined access frequency of all other files stored in the RAID set. Moreover, in video-on-demand applications, multiple users or clients could be simultaneously accessing different portions of the same continuous media file. Therefore, for high data availability, it is desirable to distribute the continuous media data for each file over all of the disk drives in each RAID set, and for load balancing, it is also desirable to distribute the parity information for each continuous media file over a multiplicity of the disk drives.

Brief Summary Text (13):
To solve these problems, the storage system uses a relatively large transfer unit including multiple data blocks when accessing a continuous media file, yet distributes the data of each continuous media file over all of the disk drives in the RAID set. Moreover, to provide load balancing, the parity associated with the data of each continuous media file is distributed across a multiplicity of disk drives in the RAID set, and preferably is distributed across all of the disk drives in the RAID set which store parity information.

Brief Summary Text (14):
In a preferred embodiment, storage in the RAID set is subdivided into a multiplicity of predefined stripe sets, and a respective number of the stripe sets are allocated to each continuous media file. Contiguous data in each stripe set is grouped into transfer blocks of multiple data blocks stored in respective individual disk drives, and each transfer block is associated with parity stored in a single one of the disk drives in the RAID set. For each disk drive in the RAID set that stores parity, the stripe set includes a transfer unit of data that is from each other disk drive in the RAID set and that is associated with that parity. Moreover, to simplify parity computation, the data in each stripe set is contiguous over the set of transfer blocks associated with the parity in each disk drive of the RAID set that includes parity.

Brief Summary Text (15):
In a specific embodiment, the RAID set includes n disk drives. The data storage of each disk drive in the RAID set is partitioned into an integer number m of hyper-volumes, and the parity is stored in one hyper-volume of each of m disk drives in the RAID set. The transfer unit includes an integer number j of data blocks, and each hyper-volume includes an integer number k of transfer units. Each stripe set includes (m)(n-1) transfer units of data. The transfer units of the RAID set are allocated for the storage of continuous media data in a right-to-left and then top-to-bottom order in which the transfer units appear in an m row by n column matrix

in which the rows of the matrix represent parity groups of hyper-volumes in the disk drives and the columns of the matrix represent the storage in the respective disk drives. For example, the RAID set includes eight disk drives, each having a capacity of 4 gigabytes. Each disk drive is partitioned into 4 hyper-volumes. The block size is 512 bytes, and the transfer unit is 256 blocks; i.e., 128 K bytes.

Brief Summary Text (16):
Because parity groups are formed of contiguous transfer units in each stripe set, at most one write access to each parity hyper-volume need be performed during write access to a stripe set. For example, for each parity group, old data from a first transfer unit is read from disk storage and exclusive-ORed with new data for the first transfer unit to produce a transfer unit of parity changes. The parity changes are stored in non-volatile memory, and then the new data for the first transfer unit is written to disk storage. Old data from a next transfer unit in the parity group is read from disk storage, exclusive-ORed with new data from the next transfer unit to produce a transfer unit of parity changes, the parity changes are exclusive-ORed with the parity changes in non-volatile memory and the result is stored back in non-volatile memory, and then the new data from the next transfer unit is stored to disk memory. The transfer unit of parity for the parity group is read from disk storage. Once all of the new data for the parity group has been written to disk storage, the parity read from the parity group is exclusive-ORed with the parity changes in the non-volatile storage to compute the new parity, the new parity is written to non-volatile storage, then the new parity is written to disk storage, and finally the non-volatile memory is deallocated.

Drawing Description Text (23):
FIG. 21 is a memory map of a RAID set including eight disk drives;

Detailed Description Text (4):
Turning now to FIG. 1 of the drawings, there is shown a video file server generally designated 20 incorporating the present invention. The video file server 20 includes an array of stream servers 21, at least one control server 28, 29, a cached disk array storage subsystem 23, and an optional tape silo 24. The video file server 20 is a high performance, high capacity, and high-availability network-attached data server. It provides the ability for multiple file systems to exist concurrently over multiple communication stacks, with shared data access. It also allows multiple physical file systems to co-exist, each optimized to the needs of a particular data service.

Detailed Description Text (11):
For multi-media data transfer, the active one of the controller servers 28, 29 assigns one of the stream servers 21 to the network client 54 requesting multi-media service. The network 25, for example, has conventional switching mechanisms, such as an ATM switch 53 or arrays of cross-bar switches, that permit any one of the clients 54 to communicate with any one of the stream servers 21. The active one of the controller servers 28, 29 could assign a stream server to a network client by a protocol sending to the client the network address of the stream server assigned to send or receive data to or from the client. Alternatively, the active one of the controller servers 28, 29 could communicate with a switching mechanism such as the ATM switch 53 to establish a data link between the client and the stream server assigned to the client.

Detailed Description Text (16):
Turning now to FIG. 3, there is shown a more detailed block diagram of the cached disk array 23. The cache memory 41 is composed of dynamic RAM cards mating with a dual redundant back-plane system bus 42. The cached disk array 23 also includes micro-processor cards that mate with the back-plane system bus 42 and are programmed to function as channel directors 43 or disk directors 44. Each of the channel directors 43 is interfaced through one of a number of SCSI adapters 45 to the SCSI interface of one of the stream servers 21. Each of the disk directors 44

is interfaced through at least one of a number of disk adapters 46 connected to a string of commodity FBA (fixed-block architecture) disk drives 47. The channel directors 43 access data in the cache memory 41 in response to a request from its associated stream server. If data to be read by a channel director are not found in cache memory, one of the disk directors 44 and disk adapters 46 transfers or "stages" the data from the disk array 47 to the cache memory 41. In a background process, the disk directors 44 and disk adapters 45 also write-back data from the cache memory 41 to the disk array 47, after the channel directors write data to the cache memory 41. In addition to providing intermediate storage for the data transferred between the channel directors 43 and the disk directors 44, the cache memory 41 also provides intermediate storage for control information transferred among the channel directors and disk directors.

Detailed Description Text (21):
The software 60 includes a file system 61 for controlling transfer of data between the network 25 and the disk array (47 in FIG. 2) or tape silo (24 in FIGS. 1 and 2). A buffer cache 62 composed of part of the random-access memory of the stream servers 21 is used as a buffer for this data transfer.

Detailed Description Text (24):
Turning now to FIG. 5, there is shown a more detailed block diagram of the software structure 60. The file system 61 in FIG. 4 has been expanded into its components. These components are a common file system 71, a group of software modules providing communication between the common file system and the network, and a group of software modules providing communication between the common file system and the cached disk array 23 or tape silo 24. The common file system 71 uses the Virtual File System (VFS), which is an industry-standard back-end file system switch, to interface with the physical file systems 79. VFS translates NFS Common File System requests, and permits NFS access to CMFS movie files for editing. (The NFS Common File System Requests in themselves are translations of NFS requests to the intended physical file storage devices. NFS is one of the file access protocols 75.) The common file system 71 accesses the buffer cache 62 during data transfers between the network (25) and disk or tape storage (23, 24).

Detailed Description Text (39):
The design of the CPU scheduler is based on a combination of weighted round-robin and rate monotonic scheduling procedures. Tasks within the isochronous class are scheduled using a rate-monotonic procedure, while the real-time and general-purpose tasks are scheduled using the weighted round-robin scheme. The isochronous class is given the highest priority; that is, any task within the isochronous class always pre-empts a real-time or a general-purpose task.

Detailed Description Text (41):
The general-purpose class supports pre-emptible tasks that are suitable for low-priority background processing. In order to ensure that general-purpose tasks can always make progress, this class is granted a minimum CPU processing quantum.

Detailed Description Text (43):
The real-time class is suitable for tasks that require guaranteed throughput and bounded delay. Real-time tasks are not pre-emptible; however, a software provision is made to allow for the existence of safe "preemption windows" in which all isochronous tasks can be executed. A weight and a scheduling flag is assigned to every real-time task. The weight is used as the means to limit the amount of processing time taken by the real-time task at each invocation. The scheduling flag is used to indicate that the task has pending work and to signal the scheduler that the task needs to be invoked. The scheduling flag may be set by an interrupt service routine or a task of any class.

Detailed Description Text (46):
The isochronous class supports real-time periodic tasks that require performance

guarantees for throughout, bounded latency, and lower jitter. Low jitter reduces the amount of buffering needed at the client, which in turn improves the response time of interactive video applications. The isochronous tasks that support streams of different periods are assigned priorities (w1, w2, w3, etc.) on a rate-monotonic basis (i.e., a task with a higher frequency has a higher priority). Isochronous tasks also allow for a safe "preemption window" in which all higher priority isochronous tasks can be executed. Isochronous tasks are used to schedule periodic network transmission of audio and video frames. An isochronous task executes exactly once per period. In the preferred implementation, a single isochronous task services all client streams that have the same frame rate.

Detailed Description Text (47):
The scheduler executes isochronous tasks from a "Ready" queue 84 in which all isochronous tasks that are ready to run are arranged in order of decreasing priority (a task with the lowest period has the highest priority and resides at the head of the queue). An isochronous task is inserted in its appropriate place on the "Ready" queue 84 upon arrival. The arrival of isochronous tasks is generated by period timers. A unique periodic timer exists in the system for each distinct period among all the admitted isochronous tasks.

Detailed Description Text (48):
Whenever an isochronous task arrives, the scheduler determines whether a currently running task needs to be pre-empted. If the currently running task is a general-purpose task, it is pre-empted by the newly arrived isochronous task. If the currently running task is a real-time task, it will be pre-empted by the newly arrived isochronous task in the next "preemption window". If the currently running task is of the isochronous class, the scheduler compares its priority to that of the task currently at the head of the "Ready" queue 84. If the priority of the current task is lower, it is pre-empted at the next "preemption window" by the isochronous task from the head of the queue. The scheduler continues to execute isochronous tasks until the isochronous "Ready" queue 84 becomes empty. Whenever the queue is empty, the scheduler alternates between the real-time and general-purpose classes using a weighted round-robin scheme.

Detailed Description Text (51):
In the absence of isochronous tasks, the scheduler can provide guarantees on throughput and delay bounds for real-time tasks (this assumes that all requests destined for a real-time task generate a constant amount of work). A maximum service delay is the time it takes to complete one round of real-time tasks scheduling plus the general purpose time quantum. Let R denote this maximum service delay in steady state. Weights may be assigned to real-time tasks to allocate and guarantee bandwidth averaged over the maximum service delay, R. If W denotes the weight given to a real-time task (the number of units of this task, or requests, processed in one round), then the task's steady state throughput is (W/R) requests per unit time.

Detailed Description Text (52):
An admission control policy is employed in order to ensure that a feasible schedule exists for all the admitted tasks; that is, all the admitted tasks can be scheduled using the combination of rate monotonic and weighted round-robin scheduling procedure described above without violating any performance guarantees. The admission control policy for access to processor resources balances the needs of the three classes of tasks: throughput and maximum delay requirements of the real-time tasks, a minimum guaranteed CPU quantum for the general-purpose tasks, and the periodic deadline-sensitive nature of the isochronous tasks. The admission control policy uses a time-based admission test for rate monotonic (isochronous) tasks with an adjustment to account for tolerable delay constraints imposed by the real-time tasks, with an adjustment to account for tolerable delay constraints imposed by the real-time tasks. Let L.sub.r denote the maximum delay that can be tolerated by any of the real-time tasks. Then a feasible schedule exists for a set of n isochronous

tasks and m real-time tasks if the following two conditions hold true: ##EQU1## where C.sub.i run-time requirement of isochronous task i

Detailed Description Text (55):
r.sub.j run-time required by the real-time task j to process one request

Detailed Description Text (57):
As noted above, C.sub.i is a fixed time per execution of isochronous task i. In the second step a test must be applied to each isochronous task i to ensure that its execution requirements can be fulfilled in the presence of all higher priority isochronous tasks. The test is as follows

Detailed Description Text (61):
The second condition is expressed by the following constraint: ##EQU5## where B.sub.i is the size of the disk buffer allocated to stream i, and M is the total amount of system memory from which the disk buffers are allocated. An equivalent amount of memory is available from which network buffers are allocated. B.sub.i bytes is the amount of data transferred from disk for session i during one round of the round-robin service for the admitted streams. Strategies for choosing an appropriate size for disk buffers are discussed below.

Detailed Description Text (63):
While describing conditions 2 and 3 for the admission control, we referred to B.sub.i, the size of a disk buffer allocated to stream i, without specifying how this size is chosen. In this section we discuss two strategies for choosing the disk buffer sizes, which is equivalent to determining the amount of data that should be transferred from the disk for each session during one round.

Detailed Description Text (64):
The "optimal strategy" is the one in which the amount of data transferred from disk for each stream is proportional to the stream's playback rate. The constant of proportionality is the disk service time for one round. The strategy is described as follows. Let M bytes denote the total amount of system memory from which the disk buffers are allocated for all streams. Then the maximum time taken by one round of filling up the disk buffers of all the streams is ##EQU7## where D.sub.min is the same as in equation (2). T is used as the constant of proportionality for sizing the disk buffers. The rate at which buffers are filled is (.SIGMA.B.sub.i)/T . The rate at which network buffers are drained is .SIGMA.R.sub.i. The simple constraint therefore is (.SIGMA.B.sub.i)/T.gtoreq..SIGMA.R.sub.i. This is simplistically satisfied for each stream if B.sub.i =T R.sub.i, where B.sub.i is the size of the disk buffer and the size of the disk read block for stream i, and R.sub.i is the stream's playback rate.

Detailed Description Text (71):
A simple case is the one in which the playback of each stream occurs at a constant bit-rate. This situation arises when the video is recorded in its original uncompressed form (frame sizes are constant) or when the video is compressed at a constant bit-rate (MPEG I, for example). In the case of a constant playback rate, all real-time disk requests may be issued to the disk exactly at the beginning of every interval of length T (T is the worst case round-robin service time as computed in the previous section). Let k denote the number of active real-time streams. Then the number of real-time requests that may be issued to the disk every T period is n-k, where n is the maximum number of streams supported by the system, as was described in the previous section. The non real-time requests may be issued at any time within the interval T, as long as the round time to service k real-time streams plus the data transfer time of the non real-time requests does not exceed T.

Detailed Description Text (79):

It may also be possible to accommodate non real-time requests simply by using two priority queues: a low priority for non real-time requests and a high priority for real-time requests. In order for such a scheme to work correctly, it is necessary to implement the priority queues at all levels including the lowest level that maintains queued disk requests, such as the disk adapter or the driver level. This scheme also requires that some portion of the disk bandwidth be reserved for non real-time requests.

Detailed Description Text (85):
Turning now to FIG. 11, there is shown a flowchart of a video prefetch routine performed by the cached disk array in response to a video prefetch command from a stream server. The video prefetch routine ensures that data specified by the video prefetch command will be in the cache of the cached disk array at the time that the cached disk array receives a subsequent fetch command from the stream server. The execution of a video prefetch routine differs from a conventional cached disk array synchronous prefetch operation by ensuring that the video prefetch routine is executed on a high priority basis, and by ensuring that the prefetched video data are retained in the cache of the cached disk array until the subsequent prefetch command is serviced.

Detailed Description Text (89):
Turning now to FIG. 12, there is shown a flowchart of a video fetch routine executed by a channel director (43 in FIG. 3) of the cached disk array in response to a video fetch command from a stream server. In a first step 131, the channel director identifies the next track in the video segment to be fetched. Then in step 132, the channel director accesses the directory in the cache memory (41 in FIG. 3) to determine whether data of the track is in the cache and to determine the cache slot containing the data of the track. If the track is not in the cache, then presumably an error has occurred, because each video fetch command specifying a video segment should have been preceded by a video prefetch command specifying the same video segment, and the video prefetch command should have been executed prior to receipt of the video fetch command. Otherwise, in step 133, the data of the track are transferred from the cache slot to a channel director buffer. Next, in step 134, the data are transferred from the channel director buffer to the stream server having issued the fetch command, and in step 135, the process of the stream server having issued the fetch command is removed from the wait list for the cache slot.

Detailed Description Text (112):
In step 173, the admission control program sets an index to point to a first valid or operable one of the stream server PCs. Then in step 174, the admission control program checks whether the desired starting time or position in the movie of the new request falls in the RAM window of the requested movie in the indexed stream server PC. If so, then in step 175 the admission control program checks whether the indexed stream server PC has sufficient resources to handle the client request. The indexed stream server PC, for example, may not have a free network link that would be needed for satisfying the request. In general, a stream server PC has a total buffer memory capacity limitation and a number of bandwidth limitations. The bandwidth limitations include a network bandwidth limitation, a throughput or buffer bandwidth limitation, and a bus bandwidth limitation for communication with the cached disk array 23 and the tape silo 34. The throughput or buffer bandwidth limitation is dependent on the type of bus used in the stream server PC. An advantage of the method used in FIG. 17 is that the throughput is used efficiently. Very little of the throughput is used for maintaining the sliding window in RAM, so that most of the throughput can be used for transmitting data from the sliding window to network clients.

Detailed Description Text (123):
Because the cached disk array 23 may use a nonvolatile write buffer and well-known RAID techniques of error correction to recover from disk drive failures, the cached

disk array can acknowledge completion of a backup operation as soon as the data are written to the cached disk array. The actual writing to tape could be done as a background process, mainly during off-peak hours, when the stream servers are not heavily loaded by data transfers to and from network clients. The cached disk array can provide "instant" restore service for backup files maintained in the cached disk array. The cached disk array can also provide temporary batch backup, without writing to tape, pending success or failure of transactions by clients that employ transactional semantics or transaction processing.

Detailed Description Text (124):
Turning now to FIG. 20, there is shown a block diagram illustrating the distribution of software used in the file server (20 in FIG. 1) for the "on-line" tape backup operations of FIG. 19. The backup software used for this purpose can be designed and written specifically for it, or it can be a modification of an existing backup package, as described below. In particular, an existing implementation of suitable backup software is adapted from the Epoch (trademark) backup software sold by EMC Corporation, 171 South Street, Hopkinton, Mass. 01748. The backup software includes a backup scheduler 201, a volume manager 202, and a save/restore data mover 203. The backup software in the file server (20 in FIG. 1) is adapted from the Epoch (trademark) Hierarchical Storage Management (HSM) software by splitting the save/restore data mover 203 from the backup scheduler 201 and volume manager 202 so that the data mover 203 can run in the environment of a separate computer. The backup scheduler 201 and the volume manager 202 comprise the "control" part of the Epoch (trademark) backup software. The backup scheduler 201 and the volume manager 202 run in the active controller server (28 or 29 in FIG. 2) to provide backup scheduling, migration and catalog management. Alternatively, the backup scheduler 201 and the volume manager 202 could run in a separate external computer (not shown), which could communicate with the stream servers 21 over a network different from the internal Ethernet 26. The save/restore data mover 203 is replicated in each of the stream servers 21, only one of which is shown in FIG. 20. The save/restore data mover 203 is responsive to commands transmitted by the backup scheduler 201 and volume manager 202 over the internal Ethernet link 26. The backup scheduler 201, the volume manager 202, and the save/restore data mover 203 are adapted to communicate via the commands over the Ethernet link 26 instead of the procedure calls that are used in the Epoch (trademark) backup software.

Detailed Description Text (129):
VII. Configuration, Access, and RAID Striving of Data Storage for Continuous Media.

Detailed Description Text (130):
The video file server preferably uses RAID techniques in order to ensure data availability and in particular to recover from a failure of any single disk drive in a RAID set. The preferred configuration of data and parity in a RAID set is shown in FIG. 21. The RAID set 220 includes a group of n disk drives, and as shown n is eight. The RAID set 220 is one of a number of RAID sets formed by grouping the disk drives in such a way that one disk drive in the set stores parity information for data in the other disk drives in the RAID set. Moreover, the parity information for the data in the RAID set is spread across more than one of the disk drives, so that access to the parity does not become a bottleneck for write operations, and multiple writes to parity and data can proceed concurrently.

Detailed Description Text (131):
FIG. 21 depicts a memory map for each of the disk drives D0, D1, D2, . . . , D7. The memory maps for the disk drives form columns in a memory map array. The rows in the memory map array are parity groups. In particular, data storage of each disk drive D0, D1, D2, . . . , D7 in the RAID set is partitioned into an integer number m of hyper-volumes H0, H1, H2, H3, and the parity is stored in one hyper-volume of each of m disk drives in the RAID set. As shown, m is 4, and therefore there are four hyper-volumes of parity designated as P1, P1, P2, and P3. Each byte of parity

in each of the parity hyper-volumes is computed as the exclusive-OR of the corresponding data bytes having the same address in each of the other disk drives in the RAID set. In FIG. 21, for example, the addresses increase consecutively from top to bottom in the memory map for each disk drive D0, D1, D2, . . . , D7.

Detailed Description Text (132):
Because the parity information in the RAID set 220 is distributed over the four disk drives D0, D1, D2, D3, the loading on the disk drives is more balanced during write access. A write access requires data to be written to at least one of the disk drives and to the disk drive for storing the associated parity information. If only one of the disk drives stored all of the parity information, then this one disk drive would need to be accessed during every write operation, which could become a bottleneck to data availability during a heavy concentration of write operations to the RAID set.

Detailed Description Text (133):
The storage of continuous media data in the RAID set introduces another problem of data availability that is particularly severe in the case of a single disk drive failure mode of operation. In such a single disk failure mode of operation, it is necessary to access all of the operational disk drives in the RAID set to reconstruct the data in the failed disk drive from the data and parity in the parity group of the reconstructed disk drive. If all of the data for any one of the continuous media files were stored in just one of the disk drives, and that one continuous media file had a very high access frequency, then all of the operational disk drives would be very heavily loaded during reconstruction of the data in the failed disk drive, possibly causing a bottleneck to the access of other files stored in the RAID set.

Detailed Description Text (134):
Unfortunately, it is very likely that a particular one of the continuous media files will have a very high access frequency in comparison to all other continuous media files stored in the RAID set. For example, the most popular new-release movie in a video-on-demand application may very well have an access frequency exceeding that of all other current movies. A continuous media file such as a popular movie also tends to be rather large and tends to be accessed concurrently by multiple users at different locations in the file. All of these circumstances tend to cause a data availability problem among the disk drives in the RAID set unless the data and the parity for each continuous media file is distributed over all of the disk drives in the RAID set.

Detailed Description Text (135):
As shown in FIG. 21, data for one continuous media file, shown with cross-hatching, are striped across all of the disk drives in the RAID set and have associated parity in each of the disk drives that store parity in the RAID set. Such striping across all of the disk drives in the RAID set, in combination with a relatively fixed overhead for disk access and the sequential or isochronous fashion of data access to the multimedia file, however, leads to a problem of a high rate of disk access unless a relatively large transfer unit of continuous media data is accessed during each disk access.

Detailed Description Text (136):
Preferably, the transfer unit includes an integer number j of data blocks, and each hyper-volume includes an integer number k of transfer units. Each stripe set includes (m)(n-1) transfer units of data, or a total of (j)(k)(m)(n-1) data blocks. For example, the transfer unit includes 256 blocks, and each block includes 512 bytes, so that the transfer unit is 128 K bytes. For a RAID set including eight disk drives and four hyper-volumes per drive, one stripe set includes 28 transfer units, or 14,336 blocks total, or 3.5 M bytes total.

Detailed Description Text (137):

Preferably, the transfer units of the RAID set are allocated for the storage of continuous media data in a right-to-left and then top-to-bottom order in which the transfer units appear in an m row by n column matrix in which the rows of the matrix represent parity groups of hyper-volumes in the disk drives and the columns of the matrix represent the storage in the respective disk drives. For example, video data appearing in the first transfer unit 221 of the stripe set shown in FIG. 21 is the earliest video data in the stripe set, and video data appearing in the last transfer unit 222 of the stripe set shown in FIG. 21 is the latest video data in the stripe set. As continuous media data is read from the stripe set in a more or less isochronous fashion, this allocation scheme staggers the timing of the individual access operations to the disk drives in the RAID set to ensure data availability despite wide variations in access frequency. The staggered timing also permits efficient parity computations during write access and only a single write access is needed to each parity hyper-volume during a write access to the entire stripe.

Detailed Description Text (138):
Preferably the respective transfer units in a stripe are accessed sequentially by incrementing a transfer unit index, and indexing a transfer unit mapping table such as the table 230 illustrated in FIG. 22 to determine the disk drive and the particular hyper-volume in the storage of the disk drive that stores the transfer unit. The first column of the table 230 is the transfer unit index, the second column of the table is a disk drive index, and the third column of the table is a hyper-volume index. For the particular organization shown in FIG. 21, the parity associated with any transfer unit is found in the disk drive identified by the hyper-volume index. In other words, the parity P0 associated with the first hyper-volume H0 of data in any drive is found in the first hyper-volume H0 in the first disk drive D0, the parity P1 associated with the second hyper-volume H1 of data in any drive is found in the second hyper-volume H1 in the second disk drive D1, etc. Therefore a single look-up in the table 230 gives the disk drive and hyper-volume containing the associated parity as well as the disk drive and hyper-volume containing the transfer unit data.

Detailed Description Text (140):
Each stripe set has an identification number consisting of a RAID set number and an index of the stripe set in the RAID set. The RAID set number identifies a particular one of a number of RAID sets or disk drive groups in the storage system. The stripe set index identifies a particular one of a number of stripe sets in the RAID set, and is used as an index for addressing a transfer unit in each disk drive in the RAID set. For example, if the transfer unit contains j blocks, and the hyper-volume contains k transfer units, then the starting block address of the transfer unit specified by the transfer unit index in the disk drive identified by the drive index from the table above is computed as: ((HVI)(k)+(SSI))(j), where HVI is the hyper-volume index from the table above and SSI is the stripe set index.

Detailed Description Text (142):
In a first step 241 of FIG. 23, a transfer unit index (tui) is set to zero, and a transfer unit counter (ntu) is set to the value of the parameter NTU. Since stripe sets are the minimum granularity of disk storage allocated to continuous media data, a first transfer unit of data can always be written to a transfer unit of disk storage for which the transfer unit index has a value of zero. Next, in step 242, execution returns if the transfer unit counter (ntu) has a value of zero. Otherwise, execution continues to step 243.

Detailed Description Text (143):
In steps 243 to 245, a value is computed for a variable (stu) indicating the value that the transfer unit counter (ntu) should have upon completing the writing of a next parity group of transfer units. In particular, in step 243, the value of the transfer unit counter is compared to the number of disk drives in the RAID set to determine whether or not the write operation can be completed by writing to a

single parity group of transfer units (i.e., to at most one transfer unit in each of the disk drives in the RAID set). If the value of the transfer unit counter is not greater or equal to the number (n) of disk drives in the RAID set, then the write operation can be completed by writing to a single parity group of transfer units, and the write operation is completed when the transfer unit counter reaches a value of zero. In this case, execution branches from step 243 to step 244 to set stu to zero. Otherwise, if the value of the transfer unit counter is not greater or equal to the number (n) of disk drives in the RAID set, then the write operation cannot be completed by writing to a single parity group of transfer units, and the write operation to the next parity group is completed when the transfer unit counter reaches a value of ntu--n+1. In this case, execution continues from step 243 to step 245 to set stu to a value of ntu--n+1. After step 244 or step 245, execution continues to step 246.

Detailed Description Text (144):
In step 246, the transfer unit mapping table (FIG. 22) is indexed with the transfer unit index (tui) to fetch the associated drive index (DI) and hyper-volume index (HVI). Then, in step 247, a starting block address (SBA) for the current transfer unit (and all other transfer units in the same parity group) is computed as SBA.rarw.((HVI)(k)+SSI)(j), where k is the number of transfer units in each hyper-volume, and j is the number of blocks in each transfer unit. Then, in step 248, data of the new transfer unit having the index (tui) are loaded into a non-volatile parity buffer. Execution continues to step 249.

Detailed Description Text (145):
Steps 249 to 255 comprise a processing loop that is traversed once for each transfer unit of data to be written to the same parity group containing at most one transfer unit of data in each disk drive of the RAID set. In step 249, the old version of the transfer unit of data indexed by tui is read from the disk drive indexed by DI beginning at the starting block address SBA. Next, in step 250, the old version read from disk is added to the parity buffer using an exclusive-OR operation in order to compute parity changes due to the writing of the new version over the old version, and in order to accumulate these parity changes in the parity buffer.

Detailed Description Text (146):
In step 251, the new version of the transfer unit of data indexed by tui is written to the disk drive indexed by DI beginning at SBA. In step 252, the transfer unit counter ntu is decremented by one, and the transfer unit index is incremented by one. Next, in step 253, the value of the transfer unit counter ntu is compared to the stopping value stu.

Detailed Description Text (147):
If the value of the transfer unit counter ntu is not less than or equal to the stopping value stu, then execution continues from step 253 to step 254 in order to index the transfer unit table with the transfer unit index tui to get the drive index (DI). Then in step 255, new data for the transfer unit indexed by tui is added into the parity buffer using an exclusive-OR operation, and execution loops back to step 249 to perform another iteration for this next transfer unit.

Detailed Description Text (149):
In step 256 of FIG. 24, a transfer unit of parity for the current transfer unit parity group is read from the disk drive HVI, beginning at the starting block address SBA last computed in step 247 of FIG. 23. Next, in step 257, execution waits for completion of the writing of data for the current transfer unit parity group, if writing has not been completed when step 257 is reached. Then, in step 258, the transfer unit of parity is added to the parity buffer, by performing an exclusive-OR of the transfer unit of parity read from the drive HVI with the parity information in the parity buffer, and storing the result back in the parity buffer. In step 259, the transfer unit is written from the parity buffer back to the drive

HVI, starting at the starting block address SBA. Finally, in step 260, the parity buffer is deallocated when the writing of the parity to the drive HVI has been completed. After step 260, execution returns to step 242 of FIG. 23. Eventually, the transfer unit counter ntu reaches a value of zero when all of the transfer units have been written to the disk drives, and execution returns from step 242 of FIG. 23.

Detailed Description Text (158):
The stream server 21 maintains a pointer 323 to the current play position in the client's current clip 321, and streams continuous media data to the client 54 from the current stripe set 324 in cache 41 of the cached disk array 23. The stream server 21 issues a prefetch command to the cached disk array so that the next stripe set 325 is prefetched from the RAID set 220 and stored in the cache 41. The transfer unit mapping table 230 for the RAID set is stored in each disk director 44 that may access the RAID set. Although not shown in FIG. 28, at least two disk directors may access each RAID set over a respective redundant link from each of the disk directors to the RAID set to provide access in case one disk director should become inoperative.

Detailed Description Text (272):
In view of the above, there has been described a way of striping a sequence of continuous media data across parity groups in a RAID set in order to provide high data availability and load balancing. The striping distributes the sequence of continuous media data across all of the disk drives in the RAID set. The sequence of continuous media data is distributed across parity groups including parity in the disk drives in the RAID set that store parity, and in each of the parity groups, across the disk drives containing data. Because the sequence of continuous media data is comprised of contiguous transfer units in each parity group, at most one write access to the parity in each parity group need be performed during write access to each parity group.

Detailed Description Paragraph Table (6):
VRPopenres.sub.-- t*vrp.sub.-- openplay.sub.-- 1(playargs.sub.-- t*,CLIENT *) where structplayargs.sub.-- t { ticket.sub.-- t ticket; ulong.sub.-- t cbprog; ulong.sub.-- t cbvers; endpoint.sub.-- t destination; cliplist.sub.-- t *clips; service.sub.-- t service; failover.sub.-- t failover; bool loop; }; ticket access control code cbprog, cbvers RPC program and version to be used for callback destination destination network address and protocol information clips list of clips to play service type of service requested: struct service.sub.-- t { flags.sub.-- t flags; priority.sub.-- t priority; ulong.sub.-- t keepalive; }; where flags requested stream features: const FLAG.sub.-- PAUSEONLY = 0x01; const FLAG.sub.-- VCRCONTROLS = 0x02; const FLAG.sub.-- AUTOCLOSE = 0x04 const FLAG.sub.-- KEEPALIVE = 0x08; const FLAG.sub.-- NOARCHIVEWAIT = 0x10; priority stream priority keepalive keepalive timer interval (in seconds) failover failover capability requested (further described below) loop TRUE/FALSE if clips should loop forever Returns: struct VRPopenres.sub.-- t { VRPstatus.sub.-- t status; sHandle.sub.-- t handle; endpoint.sub.-- t source; }; status status code handle streamhandle if successful source source endpoint allocated if successful

Other Reference Publication (2):
Martin E. Schulze, "Considerations in the Design of a RAID Prototype," Computer Science Division, EECS, University of California at Berkeley, CA, Aug. 25, 1988.

Other Reference Publication (4):
Gray et al., "Parity Striping of Disc Arrays: Low-Cost Reliable Storage with Acceptable Throughput," Technical Report 90.2, Tandem Computers, Cupertino, CA., Jan. 1990.

Other Reference Publication (5):

h e b b g eeef c e h e ge

Edward K. Lee, "Software and Performance Issues in the Implementation of a RAID Prototype," Computer Science Division, EECS, University of California at Berkeley, CA, Mar. 1989.

Other Reference Publication (9):
Patterson et al., "A Case for Redundant Arrays Of Inexpensive Disks (RAID)," Report No. UCB/CSD 87/391, Computer Science Division (EECS), University of California, Berkeley, California, Dec. 1987, pp. 1-24.

Other Reference Publication (10):
Patterson et al., "Introduction to Redundant Arrays of Inexpensive Disks (RAID)," COMPCON 89 Proceedings, Feb. 27-Mar. 3, 1989, IEEE Computer Society, pp. 112-117.

Other Reference Publication (35):
Federighi C, "A Distributed Hierarchical Storage Manager for a Video-on-Demand System," Department of Electrical Engr. and Computer Science, University of California, Berkeley, California, Dec. 1993.

CLAIMS:

1. A method of striping a sequence of continuous media data across a plurality of n disk drives in a RAID set storing data and associated parity across the disk drives, wherein the data storage of each disk drive in the RAID set is partitioned into an integer number m of hyper-volumes, parity is distributed among the disk drives in the RAID set and is stored in at least one hyper-volume of each of m disk drives in the RAID set, and transfer units of data storage in the n disk drives are associated with the sequence of continuous media data in a right-to-left and then top-to-bottom order in which the transfer units appear in an m row by n column matrix in which the rows of the matrix represent parity groups of hyper-volumes in the disk drives and the columns of the matrix represent the data storage in the respective n disk drives in the RAID set, which further includes allocating data storage in the RAID set for the stream of continuous media data in fixed size stripe sets, each stripe set comprising a segment of the sequence of continuous media data distributed across each of the disk drives and each of the parity groups of hyper-volumes in the disk drives,

wherein the transfer unit has a predetermined size, and each stripe set includes (m)(n-1) transfer units of data.

2. A method of striping a sequence of continuous media data across a plurality of n disk drives in a RAID set storing data and associated parity across the disk drives, wherein the data storage of each disk drive in the RAID set is partitioned into an integer number m of hyper-volumes, parity is distributed among the disk drives in the RAID set and is stored in at least one hyper-volume of each of m disk drives in the RAID set, and transfer units of data storage in the n disk drives are associated with the sequence of continuous media data in a right-to-left and then top-to-bottom order in which the transfer units appear in an m row by n column matrix in which the rows of the matrix represent parity groups of hyper-volumes in the disk drives and the columns of the matrix represent the data storage in the respective n disk drives in the RAID set, which further includes allocating data storage in the RAID set for the stream of continuous media data in fixed size stripe sets, each stripe set comprising a segment of the sequence of continuous media data distributed across each of the disk drives and each of the parity groups of hyper-volumes in the disk drives, and which includes associating a name of a continuous media data file with a list of the stripe sets.

4. A method of accessing a continuous media data file including continuous media data stored in stripe sets in a RAID set of disk drives, said method comprising the steps of:

a) accessing a directory of file names of continuous media data files to locate a list of the stripe sets associated with a name of the continuous media data file;

b) accessing the list of stripe sets to determine a sequence of the stripe sets storing the continuous media data; and

c) accessing the RAID set of disk drives to retrieve the sequence of stripe sets.

5. The method as claimed in claim 4, wherein the list of stripe sets includes at least one entry specifying a starting stripe set number and an ending stripe set number, and said method includes accessing the RAID set of disk drives to retrieve a stripe set identified by the starting stripe set number, to retrieve stripe sets identified by stripe set numbers between the starting stripe set number and the ending stripe set number, and to retrieve a stripe set identified by the ending stripe set number.

6. The method as claimed in claim 4, wherein the stripe set is comprised of a series of transfer units in disk drives of the RAID set, and said method includes the step of sequentially advancing an index and accessing a look-up table with the index in order to determine a disk drive in the RAID set containing a next transfer unit to be accessed.

7. A data storage system comprising:

a plurality of n disk drives arranged in a RAID set; and

a controller coupled to the disk drives for accessing a sequence of continuous media data stored in the disk drives;

wherein the sequence of continuous media data and associated parity is striped across the disk drives in the RAID set, the data storage of each disk drive in the RAID set is partitioned into an integer number m of hyper-volumes, parity is distributed among the disk drives in the RAID set and is contained in at least one hyper-volume of each of the m disk drives in the RAID set, and transfer units of data storage in the n disk drives are associated with the sequence of continuous media data in a right-to-left and then top-to-bottom order in which the transfer units appear in an m row by n column matrix in which the rows of the matrix represent parity groups of hyper-volumes in the disk drives and the columns of the matrix represent the data storage in the respective n disk drives in the RAID set,

wherein data storage in the RAID set for the stream of continuous media data is allocated in fixed size stripe sets, each stripe set comprising a segment of the sequence of continuous media data distributed across each of the disk drives and each of the parity groups of hyper-volumes in the disk drives, and wherein the transfer unit has a predetermined size, and each stripe set includes (m)(n-1) transfer units of data.

8. A data storage system comprising:

a plurality of n disk drives arranged in a RAID set; and

a controller coupled to the disk drives for accessing a sequence of continuous media data stored in the disk drives;

wherein the sequence of continuous media data and associated parity is striped across the disk drives in the RAID set, the data storage of each disk drive in the RAID set is partitioned into an integer number m of hyper-volumes, parity is distributed among the disk drives in the RAID set and is contained in at least one hyper-volume of each of the m disk drives in the RAID set, and transfer units of data storage in the n disk drives are associated with the sequence of continuous

media data in a right-to-left and then top-to-bottom order in which the transfer units appear in an m row by n column matrix in which the rows of the matrix represent parity groups of hyper-volumes in the disk drives and the columns of the matrix represent the data storage in the respective n disk drives in the RAID set, wherein data storage in the RAID set for the stream of continuous media data is allocated in fixed size stripe sets, each stripe set comprising a segment of the sequence of continuous media data distributed across each of the disk drives and each of the parity groups of hyper-volumes in the disk drives, and wherein the controller has a directory associating a name of a continuous media data file with a list of the stripe sets.

10. A data storage system comprising:

a plurality of disk drives arranged in a RAID set; and

a controller coupled to the disk drives for accessing a sequence of continuous media data stored in stripe sets in a RAID set of disk drives, the controller having a directory of continuous media data files and lists of the stripe sets associated with each of the continuous media data files, the controller being programmed to respond to a request to access a specified continuous media data file by accessing the directory to find a list of stripe sets associated with the specified continuous media data file, to access the list of stripe sets to determine a sequence of the stripe sets storing the continuous media data, and to access the RAID set of disk drives to retrieve the sequence of stripe sets.

11. A data storage system comprising:

a plurality of disk drives arranged in a RAID set; and

a controller coupled to the disk drives for accessing a sequence of continuous media data stored in stripe sets in a RAID set of disk drives, the controller having a directory of continuous media data files and lists of the stripe sets associated with each of the continuous media data files, the controller being programmed to respond to a request to access a specified continuous media data file by accessing the directory to find a list of stripe sets associated with the specified continuous media data file, to access the list of stripe sets to determine a sequence of the stripe sets storing the continuous media data, and to access the RAID set of disk drives to retrieve the sequence of stripe sets, wherein the list of stripe sets includes at least one entry specifying a starting stripe set number and an ending stripe set number, and the controller is programmed to access the RAID set of disk drives to retrieve a stripe set identified by the starting stripe set number, to retrieve stripe sets identified by stripe set numbers between the starting stripe set number and the ending stripe set number, and to retrieve a stripe set identified by the ending stripe set number.

12. A data storage system comprising:

a plurality of disk drives arranged in a RAID set; and

a controller coupled to the disk drives for accessing a sequence of continuous media data stored in stripe sets in a RAID set of disk drives, the controller having a directory of continuous media data files and lists of the stripe sets associated with each of the continuous media data files, the controller being programmed to respond to a request to access a specified continuous media data file by accessing the directory to find a list of stripe sets associated with the specified continuous media data file, to access the list of stripe sets to determine a sequence of the stripe sets storing the continuous media data, and to access the RAID set of disk drives to retrieve the sequence of stripe sets, and wherein the stripe set is comprised of a series of transfer units in disk drives of the RAID set, and the controller is programmed to sequentially advance an index and

access a look-up table with the index in order to determine a disk drive in the RAID set containing a next transfer unit to be accessed.

First Hit Fwd Refs
End of Result Set

Generate Collection Print

L16: Entry 2 of 2 File: USPT Feb 4, 2003

DOCUMENT-IDENTIFIER: US 6516425 B1
TITLE: Raid rebuild using most vulnerable data redundancy scheme first

Abstract Text (1):
A method of managing data in a hierarchical data storage system employing data redundancy schemes includes prioritizing a data rebuild based on a most vulnerable data redundancy scheme in the storage system. A data storage system embodying this rebuild prioritization method is also described. Prioritizing the data rebuild includes enabling a rebuild of the most vulnerable data redundancy scheme prior to enabling a rebuild of any other data redundancy scheme in the system. The most vulnerable data redundancy scheme is determined by comparing a probability of losing data that can be prevented by a rebuild for each data redundancy scheme with respect to the potential for one or more next storage device failures in the data storage system. The probability of losing data for each data redundancy scheme is determined by considering characteristics associated with the storage system and disk drives in the storage system, including number of storage devices, mean time between failure, mean time or calculated time to rebuild, and failure dependencies.

Brief Summary Text (2):
This invention relates in general to computer mass storage systems and, more particularly, to prioritizing data rebuild in the event of a disk failure in a hierarchical system utilizing a Redundant Array of Independent Disks (RAID).

Brief Summary Text (4):
Conventional disk array data storage systems have multiple disk storage devices that are arranged and coordinated to form a single mass storage system. A Redundant Array of Independent Disks (RAID) system is an organization of data in an array of mass data storage devices, such as hard disk drives, to achieve varying levels of data availability and system performance. Data availability refers to the ability of the RAID system to read and write data in the array of data storage devices even in the event of a failure of one or more of the individual data storage devices or components in the array. A measurement of system performance is the rate at which data can be sent to or received from the RAID system.

Brief Summary Text (6):
RAID systems typically designate part of the physical storage capacity in the array to store redundant data, either mirror or parity. The redundant information enables regeneration of user data in the event that one or more of the array's member disks, components, or the access paths to the disk(s) fail. Typically, the disks are divided into equally sized address areas referred to as "blocks." A set of blocks that has the same unit address ranges from each disk is referred to as a "stripe" or "stripe set." A set (or subset) of disks in the array over which a stripe or stripe set spans is referred to as a redundancy group. Traditionally, RAID arrays employ one or more redundancy groups and a single redundancy scheme for each redundancy group, although the schemes may vary among the redundancy groups. However, as will be discussed subsequently herein, hierarchical RAID arrays employ one or more redundancy schemes (i.e., RAID levels) for each redundancy group in an

array.

Brief Summary Text (7):
From a data management and data redundancy perspective, RAID levels are typically characterized as one of six architectures, or redundancy schemes, enumerated as RAID levels 1-6. Although other RAID levels exist, levels 1-6 are the most commonly used and will be discussed herein with respect to the present invention. However, it should be noted that the present invention is applicable to any RAID level or data redundancy scheme.

Brief Summary Text (8):
The use of disk mirroring is referred to as RAID Level 1, where original data is stored on one set of disks and a duplicate copy of the data is kept on separate disks. The use of parity checking is referred to as RAID Levels 2, 3, 4, 5, and 6. In general, although RAID 1 provides higher data reliability and may provide the best small-write input/output (I/O) performance over RAID Levels 2, 3, 4 and 5, it uses the most storage space because all data is duplicated. In contrast, RAID Levels 2-5 provide a lesser amount of data reliability (relative to RAID 1) and, typically, reduced small-write performance. However, they don't consume as much disk space as a RAID 1 technique because data is not duplicated but rather interleaved and parity checked across the disk array in a stripe set. A parity stripe set interleaves data and redundant (parity) data on multiple member disks. The parity stripe set presents a single virtual disk whose user data capacity is approximately the sum of the capacities of its members, less the storage used for holding the parity (redundant) data of the user data. For RAID levels 3-5, parity is commonly calculated using a bit by bit Exclusive OR function of corresponding data chunks in a stripe set from all of the data disks. This corresponds to a one equation, one unknown, sum of products calculation. The mirror set in a RAID 1 architecture presents a single virtual disk whose user data capacity is the sum of the capacity of one-half of its members, the other half holding the mirrored (redundant) data of the user data.

Brief Summary Text (9):
In addition to RAID mirror level 1, RAID parity levels 4, 5 and 6 are of particular interest for the present discussion. Specifically, for example, a RAID 4 uses a stripe set or redundancy group and a single dedicated parity disk to store redundant information about the data existing on the other data disks in the stripe set or redundancy group. Segments of data from each virtual disk sector are distributed across corresponding sectors of all but one of the stripe set members (i.e., the parity disk), and the parity of the distributed segments is written in the corresponding sector of the parity disk.

Brief Summary Text (10):
Because a RAID 4 system stores all parity blocks on a single unit in the stripe set, the single unit containing the parity blocks is accessed disproportionately relative to the other data storage devices in the stripe set or redundancy group. To eliminate the resulting constriction of data flow in a RAID 4 system, a RAID 5 architecture distributes the parity blocks across all of the data storage devices in the stripe set or redundancy group. Typically in a RAID 5 system, a set of N+1 data storage devices forms the stripe set or redundancy group. Each stripe has N blocks of data and one block of parity data. The block of parity data is stored in one of the N+1 data storage devices. The parity blocks corresponding to the remaining stripes of the stripe set or redundancy group are stored across the data storage devices within the stripe set or redundancy group. For example, in a RAID 5 system using five data storage devices in a given stripe set or redundancy group, the parity block for the first stripe of blocks may be written to the fifth device; the parity block for the second stripe of blocks may be written to the fourth device; the parity block for the third stripe of blocks may be written to the third device; etc. Typically, the location of the parity block for succeeding blocks shifts to the succeeding logical device in the stripe set or redundancy group,

although other patterns may be used.

Brief Summary Text (11):
A RAID 6 architecture is similar to RAID 4 and 5 in that data is striped, but is dissimilar in that it utilizes two independent and distinct parity values for the original data, referred to here as P & Q. The P parity is commonly calculated using a bit by bit Exclusive OR function of corresponding data chunks in a stripe from all of the data disks. This corresponds to a one equation, one unknown, sum of products calculation. On the other hand, the Q parity is calculated linearly independent of P, but again using a different algorithm for sum of products calculation. As a result, each parity value is calculated using an independent algorithm and each is stored on a separate disk in the stripe set or redundancy group. Consequently, a RAID 6 system can rebuild data (assuming rebuild space is available) even in the event of a failure of two separate disks within a stripe set or redundancy group, whereas a RAID 5 system can rebuild data only in the event of no more than a single disk failure within a stripe set or redundancy group.

Brief Summary Text (12):
Similar to RAID 5, a RAID 6 architecture distributes the two parity blocks across all of the data storage devices in the stripe set or redundancy group. Thus, in a stripe set or redundancy group of N+2 data storage devices, each stripe has N blocks of data and two blocks of independent parity data. One of the blocks of parity data is stored in one of the N+2 data storage devices, and the other of the blocks of parity data is stored in another of the N+2 data storage devices. The parity blocks corresponding to the remaining stripes of the stripe set or redundancy group are stored across the data storage devices within the stripe set or redundancy group. For example, in a RAID 6 system using five data storage devices in a given stripe set or redundancy group, the parity blocks for the first stripe of blocks may be written to the fourth and fifth devices; the parity blocks for the second stripe of blocks may be written to the third and fourth devices; the parity blocks for the third stripe of blocks may be written to the second and third devices; etc. Typically, again, the location of the parity blocks for succeeding blocks shifts to the succeeding logical device in the stripe set or redundancy group, although other patterns may be used.

Brief Summary Text (13):
More information detailing the architecture and performance of RAID systems can be found in The RAID Book: A Source Book for RAID Technology, by the RAID Advisory Board, published Jun. 9, 1993, the disclosure of which is incorporated in full herein by reference. Additionally, a background discussion of RAID systems, and various ways to logically partition RAID systems, is found in U.S. Pat. No. 5,519,844 to David C. Stallmo, entitled "Logical Partitioning of a Redundant Array Storage System," incorporated in full herein by reference.

Brief Summary Text (14):
A hierarchical data storage system permits data to be stored according to one or more different techniques, such as different redundancy schemes, RAID levels, redundancy groups, or any combination of these. For example, in a hierarchical RAID system, data can be stored in one or more redundancy groups, and for each redundancy group, according to one or more RAID architectures (levels) to afford tradeoffs between the advantages and disadvantages of the redundancy techniques. For purposes of this disclosure, a hierarchical system includes: (1) one that automatically or dynamically migrates data between redundancy schemes (i.e., different RAID levels) and/or redundancy groups for optimum system tuning and performance, and (2) one that requires user input to configure redundancy schemes or groups, such as one that requires a user to designate a given RAID level with an identified set or subset of disks in a system array.

Brief Summary Text (15):
U.S. Pat. No. 5,392,244 to Jacobson et al., entitled "Memory Systems with Data

Storage Redundancy Management", incorporated in full herein by reference, describes a hierarchical RAID system that enables data to be migrated from one RAID type to another RAID type as data storage conditions and space demands change. This patent describes a multi-level RAID architecture in which physical storage space is mapped into a RAID-level virtual storage space having mirror and parity RAID areas (e.g., RAID 1 and RAID 5). The RAID-level virtual storage space is then mapped into an application-level virtual storage space, which presents the storage space to the user as one large contiguously addressable space. During operation, as user storage demands change at the application-level virtual space, data can be migrated between the mirror and parity RAID areas at the RAID-level virtual space to accommodate the changes. For instance, data once stored according to mirror redundancy may be shifted and stored using parity redundancy, or vice versa.

Brief Summary Text (16):
With data migration, the administrator is afforded tremendous flexibility in defining operating conditions and establishing logical storage units (or LUNs). As one example, this type of RAID system can initially store user data according to the optimum performing RAID 1 configuration. As the user data approaches and exceeds 50% of a stripe set or redundancy group array capacity, the system can then begin storing data according to both RAID 1 and RAID 5, and dynamically migrating data between RAID 1 and RAID 5 in a continuous manner as storage demands change. At any one time during operation, the data might be stored as RAID 1 or RAID 5 or both on all of the disks. The mix of RAID 1 and RAID 5 storage changes dynamically with the data I/O and storage capacity. This allows the system to dynamically optimize performance and available capacity versus an increasing amount of user data.

Brief Summary Text (17):
Clearly, each RAID level has characteristic cost-performance and cost-capacity ratios. Importantly, however, RAID systems maintain and manage redundant data to enable a recovery of the data in the event of a storage disk or component failure. To this regard, each RAID level also has a characteristic availability that determines the mean time to data loss as a function of the number of disks employed, i.e., different RAID levels provide different degrees of data protection. In the event of a disk or component failure, redundant data is retrieved from the operable portion of the system and used to regenerate or rebuild the original data that is lost due to the component or disk failure. Specifically, when a disk in a RAID redundancy group fails, the array attempts to rebuild data on the surviving disks of the redundancy group (assuming space is available) in such a way that after the rebuild is finished, the redundancy group can once again withstand a disk failure without data loss. Depending upon system design, the rebuild may be automated or may require user input. Design factors that affect rebuild include, for example, whether a spare disk is specified in the array, or whether the failed disk must be manually replaced by a user.

Brief Summary Text (18):
After detecting a disk or component failure and during a rebuild of data, regardless of rebuild design, the system remains subject to yet further disk or component failures in the same stripe set or redundancy group before the rebuild is complete. In any RAID system, this is significant because the vulnerability of data loss is dependent upon the RAID architecture (redundancy scheme) employed for that data.

Brief Summary Text (19):
For example, consider a hierarchical RAID array that uses ten disks in a single redundancy group with RAID 1 and RAID 5 storage schemes, where each RAID level may be employed separately or jointly on any one or more of the disks. The RAID 5 storage includes a stripe of data of a single block size on each disk. Each of nine disks holds actual user data in its respective block of the stripe. The tenth disk holds a block of data in the stripe containing redundant (parity) information. If a disk fails, the data on the failed disk can be reconstructed from the data on the

remaining nine disks. The original and reconstructed data can then be re-written across the nine remaining good disks in a new stripe (with one of those disks being designated as the parity disk for the new stripe). However, if any of the nine remaining disks fail before all of the RAID 5 data is rebuilt and re-written, then data will be lost.

Brief Summary Text (20):
For the RAID 1 storage of this example, data stored on a first of the 10 disks is mirrored only on a second disk in the array. If the first disk fails, then during rebuild from the second mirror disk, data will only be lost if the second mirror disk fails. Other disks could fail and the array would not lose the RAID 1 data, but RAID 5 data would be lost.

Brief Summary Text (21):
Therefore, in this example of RAID 1 and RAID 5 hierarchical storage, in the event of a disk failure and during a rebuild, the RAID 5 storage is more vulnerable, i.e., has a greater probability of data loss, than the RAID 1 storage. However, conventional hierarchical RAID systems rebuild data irrespective of the vulnerability of the RAID levels employed.

Brief Summary Text (22):
Accordingly, to minimize the probability of data loss during a rebuild in a hierarchical RAID system, there is a need to manage data recovery and rebuild that accounts for data availability characteristics of the hierarchical RAID levels employed.

Brief Summary Text (24):
According to principles of the present invention in a preferred embodiment, in a hierarchical data storage system employing data redundancy schemes, such as a RAID system, a method of managing data in response to a disk failure in the storage system includes prioritizing a data rebuild based on a most vulnerable data redundancy scheme identified in the storage system. Prioritizing the data rebuild includes enabling a rebuild of the most vulnerable data redundancy scheme prior to enabling a rebuild of any other data redundancy scheme in the system. The most vulnerable data redundancy scheme is determined by comparing a probability of losing data that can be prevented by a rebuild for each data redundancy scheme in the system with respect to the potential for one or more next disk failures in the data storage system. The probability of losing data for each data redundancy scheme is determined by considering characteristics associated with the storage system and storage devices in the array, such as number of storage devices, number of device failures, mean time between failure, mean time or calculated time to rebuild, and failure dependencies.

Brief Summary Text (25):
The present invention further includes a data storage system and apparatus embodying the rebuild prioritization method described.

Drawing Description Text (2):
FIG. 1 is a block diagram of a hierarchical data storage system employing one embodiment of the present invention rebuild prioritazation system and method.

Detailed Description Text (2):
FIG. 1 is a block diagram of a hierarchical RAID data storage system 10 employing one embodiment of the present invention rebuild prioritization system 35 and method for prioritizing a data rebuild in the event of a disk failure, component failure or other system failure indicative of an inability to access certain data in the system. The data storage system 10 includes a disk array 15 having a plurality of storage disks 20, a disk array controller cluster 25 coupled to the disk array 15 to coordinate data transfer to and from the storage disks 20, and hierarchical RAID management system 30. The RAID management system 30 controls data storage and the

hierarchical RAID levels (redundancy schemes) in the array 15, and controls the transferring of data among the various hierarchical redundancy schemes. Importantly, the RAID management system 30 includes a data rebuild prioritization system 35 according to the present invention. Specifically, in the event of a disk or component failure for any storage disk 20, the data rebuild prioritization system 35 minimizes potential data loss by prioritizing the rebuild according to the most vulnerable RAID level first.

Detailed Description Text (4):
The disk array controller cluster 25 is coupled to the disk array 15 via one or more interface buses 40, such as a small computer system interface (SCSI). The RAID management system 30 is operatively coupled to the disk array controller cluster 25 via an interface protocol 45. In the system shown, disk array controller cluster 25 is implemented as a dual controller consisting of disk array controller "A" 25A and disk array controller "B" 25B. Dual controllers 25A and 25B enhance reliability by providing continuous backup and redundancy in the event that one controller becomes inoperable. However, the present invention can be practiced with more than two controllers, a single controller or other architectures.

Detailed Description Text (5):
In a preferred embodiment, the RAID management system 30 and data rebuild prioritization system 35 are configured as firmware within a non-volatile random access memory (NVRAM) 25C, 25D of the disk array controller cluster 25. The RAID management system 30 and data rebuild prioritization system 35 include executable instructions, routines, tables and/or other data and data structures necessary for managing the controller cluster 25 and array 15, including rebuilding data after a disk failure, and for prioritizing and/or managing the order or sequence of rebuild under the present invention as will be discussed more fully herein. The RAID management system 30 and data rebuild prioritization system 35 are shown as separate from the controller cluster 25 simply for clarity of depiction and descriptions purposes. Alternatively, the RAID management system 30 and data rebuild prioritization system 35 are embodied as a separate component (as shown), either as firmware or software, or configured within a memory of a host computer 50. Alternatively, these systems are embodied in hardwired circuitry (such as an ASIC). The data storage system 10 is coupled to the host computer 50 via an I/O interface bus 55.

Detailed Description Text (6):
It should be noted here that this disclosure does not detail general RAID management techniques or actual data rebuild techniques in the event of a disk failure because these are already sufficiently well known and disclosed in the art. Rather, the present invention disclosure focuses on identifying implementations and methods for prioritizing and/or managing the order or sequence of rebuild of an array system employing multiple (hierarchical) data redundancy levels and/or redundancy groups. Additionally, although a RAID system is specifically shown and discussed, it will be understood that the present invention is applicable to any data redundancy schemes, techniques, groups or systems.

Detailed Description Text (7):
The hierarchical disk array 15 is characterized by different storage spaces, including its physical storage space and one or more virtual RAID level storage spaces. These views of storage are related through mapping techniques. Specifically, the physical storage space of the disk array 15 is mapped into a virtual storage space which delineates storage areas according to the various RAID data reliability levels (redundancy schemes). For example, the storage disks 20 in the disk array 15 can be conceptualized as being arranged into multiple redundancy groups 60, 65, 70, 75, each having a defined subset of the disks 20, with each redundancy group being configured into stripe sets having stripes 60A, 60B, 60C, 65A, 65B, 65C, 70A, 70B, 70C, and 75A, 75B, 75C. Four exemplary redundancy groups are shown 60, 65, 70, 75, each having stripes zero through "N", but any number of

groups or stripes are feasible under the present invention.

Detailed Description Text (8):
At any given point in time, each stripe is allocated a specific RAID level redundancy scheme and each redundancy group includes one or more RAID levels in its stripe set. In a preferred embodiment, RAID management system 30 migrates the redundancy schemes among the stripes and redundancy groups, depending upon storage capacity, data reliability and performance needs. For example, at a given point in time, stripe 0 (60A) of redundancy group 60 may be designated as RAID level 1 mirror redundancy, and stripes 1 (60B) and N (60C) may be designated as RAID level 5 parity redundancy. On the other hand, an entire redundancy group 65 may be designated as RAID level 6, and the other redundancy groups 70 and 75 may be designated as any combination of single or multiple redundancy schemes, including RAID levels 2, 3, 4, 5 or 6. Importantly, as described, each of these RAID levels may be configured on the same or separate disks or any combination thereof, all within stripe sets or redundancy groups in the disk array 15.

Detailed Description Text (9):
The data storage system 10 includes a memory map store that provides for persistent storage of the virtual mapping information used to map the disk array 15. The memory map store is external to the disk array and, preferably, redundantly resident in a memory 25C and 25D of the disk array controller cluster 25. The memory mapping information can be continually or periodically updated by the controller cluster 25 or the RAID management system 30 as the various mapping configurations among the different views change.

Detailed Description Text (10):
As indicated, the disk array 15 has multiple storage disk drive devices 20, and the management of data redundancy on the devices 20 is coordinated by the RAID management system 30. When viewed by the user or host application program, an application-level virtual view can represent a single large storage capacity indicative of the available storage space on the storage disks 20 within the array 15. As mentioned, in a preferred embodiment, the RAID management system 30 dynamically alters the configuration of the RAID areas over the physical storage space. As a result, the mapping of the RAID areas in a RAID-level virtual view onto the disks and the mapping of a front end virtual view to the RAID view are generally in a state of change. The memory map store in memories 25C and 25D maintains the current mapping information used by the RAID management system 30 to map the RAID areas onto the disks, as well as the information employed to map between the virtual views. As the RAID management system dynamically alters the RAID level mappings, it also updates the mapping information in the memory map store to reflect the alterations.

Detailed Description Text (11):
Now, importantly, in the event of any disk failure relative to any one or more of the disks 20, the data rebuild prioritization system 35 of the present invention is activated to prioritize redundancy schemes (i.e., RAID levels) in the array 15 and enable a rebuild of the data in such a manner that potential data loss is minimized. Specifically, in response to a disk failure or multiple disk failures, the present invention prioritizes the redundancy schemes in each of the redundancy groups to enable a storage rebuild of the most vulnerable RAID level first, the second-most vulnerable RAID level next, and so forth. A most vulnerable redundancy scheme (i.e., RAID level) is defined as a redundancy scheme having the highest probability for potential data loss that can be prevented by a rebuild in the array 15 (relative to any other RAID levels within its redundancy group and relative to any other RAID levels within any other redundancy group in the array, all with respect to the potential for one or more next storage device failures in the array). Where a same redundancy scheme (i.e., RAID level) is employed in two or more separate redundancy groups, then a most vulnerable redundancy scheme may be defined relative to a given redundancy group.

Detailed Description Text (12):
Advantageously, this prioritization technique is applied to the array 15 regardless of the RAID levels or other data redundancy levels employed or that may be employed. The vulnerability is measured through the probability of losing data that can be prevented by a rebuild given that a disk or multiple disks 20 has/have failed. In a preferred embodiment, the data rebuild prioritization system 35 does not actually rebuild the data that is subject to the failure but, rather, identifies and prioritizes the order of rebuild so that the RAID management system 30 may perform the rebuild.

Detailed Description Text (13):
Due to the existence of multiple RAID levels on the same disk or disks 20 in a hierarchical architecture, or due to the existence of one or more RAID levels on one or more redundancy groups, enabling a rebuild of one RAID level over another by rebuild prioritization system 35 after a disk failure occurs may reduce exposure to subsequent disk failures that may actually lead to a loss of data. In this context, the order in which RAID levels are rebuilt may be critical because the probability of losing data may be reduced.

Detailed Description Text (14):
The order or sequence of rebuild is very dependent on the actual RAID levels that are supported by the disk storage system 10 and the sequence in which a disk failure or disk failures occur. Due to the huge number of combinations of disks failures, and the RAID levels that may co-exist on a given disk array 15 or set of disks 20, the following examples will demonstrate the rebuild order (prioritization) analysis and steps performed by the data rebuild prioritization system 35 according to a preferred embodiment.

Detailed Description Text (15):
First, assume that we have a set of N disks 20 in a single redundancy group (group 1) and, for simplicity, assume N is even. Additionally, assume that all of the N disks are configured in the following RAID levels: RAID 1, where a portion of a disk is mirrored to another disk, and RAID 6, where N-2 disks contain actual data, and two disks contain two independent parities (two extra redundancy information).

Detailed Description Text (16):
Again, it should be noted that these are not the only RAID levels that can be supported by this invention; but for the purpose of demonstration, we are limiting our example to the above RAID levels.

Detailed Description Text (17):
Next, let us define conditional probability P(x,y.vertline.z) as the probability of losing RAID x (or redundancy scheme) data in redundancy group y that can be prevented by rebuild, given that z disks in redundancy group y have been identified as failed. In mathematical terms

Detailed Description Text (18):
P(x,y.vertline.z)=Pr (losing RAID x data in redundancy group y that can be prevented by rebuild .vertline.z disks in redundancy group y have been identified as failed)

Detailed Description Text (20):
Because a rebuild by rebuild prioritization system 35 does not start until a disk or multiple disk failures are encountered, the above probabilities must be conditioned on disk failure. Thus, for a single redundancy group system (y=1): P (1,1.vertline.1): Probability of losing RAID 1 data only in redundancy group 1 that can be prevented by rebuild given that one disk has failed in redundancy group 1; and P(6,1.vertline.1): Probability of losing RAID 6 data only in redundancy group 1 that can be prevented by rebuild given that one disk has failed in redundancy group

1.

Detailed Description Text (21):
Also, let MTBF correspond to disk Mean Time Between Failure, and MTTR correspond to Mean Time To Rebuild. The MTBF and MTTR characteristics are determined by conventional methods, such as using statistical samples, or obtained from disk manufacturer specifications. Although MTTR is used for this discussion, it will be understood that a calculated time to rebuild is also feasible and may be substituted for MTTR. Additionally, it is understood that the MTTR is RAID level dependent and system configuration dependent. However, for ease of discussion purposes, let us assume that the MTTR (that may include disk replacement time if no on-line spare is present) for any RAID level is constant. Also, let us assume that disk failures are random (and follow exponential distribution). Then, in this context, the steady state probability (limiting probability) of a given disk to fail during the rebuild time is calculated by:

Detailed Description Text (22):
Given the forgoing, the following scenarios will now be analyzed: Case 1: One disk failed Case 2: Two disks failed, not mirror of each other in RAID 1 Case 3: Two disks failed, mirror of each other in RAID 1 Case 4: Three disks or more failed.

Detailed Description Text (23):
Case 1: RAID 1 and RAID 6; One Disk Failed

Detailed Description Text (24):
For Case 1, we have one disk failed, and if the next disk failure happens to be the mirror image of the failed disk in RAID 1 while the failed disk is still being rebuilt, then RAID 1 data loss will occur. It follows: ##EQU1##

Detailed Description Text (25):
On the other hand, if the next disk failure is not a mirror of the failed disk in RAID 1 (any disk, as long as it is not a mirror of the first failed disk, N-2 possible ways), then no data loss in RAID 6 will occur. It is only the failure of a third disk (any surviving disk, N-2 possible ways) that will cause the RAID 6 data loss. Thus, ##EQU2##

Detailed Description Text (26):
Consequently, P(1,1.vertline.1) and P(6,1.vertline.1) are compared to determine which RAID level should be prioritized to be rebuilt first. Clearly, the higher the probability, the higher the priority for that RAID level to be rebuilt first. Accordingly, for most Case 1 scenarios, RAID 1 is the most vulnerable to potential data loss and thus will be prioritized to be rebuilt first. Notably, an exception exists to rebuild RAID 6 first when N becomes large. For example, RAID 6 is rebuilt first for this case study when: ##EQU3##

Detailed Description Text (27):
Case 2: RAID 1 and RAID 6; Two Disks Failed, not Mirror of Each Other

Detailed Description Text (28):
For Case 2, if the two disks that have failed are not a mirror of each other in RAID 1, then a next disk failure (any surviving disk, N-2 possible ways) during the rebuild will cause a data loss in RAID 6. Thus: ##EQU4##

Detailed Description Text (29):
However, RAID 1 data can be lost too if it happens that the third disk failure is a mirror of one of the other two failed disks. The third disk must be a mirror of the first failed disk or a mirror of the second failed disk in RAID 1 (there are two possible ways), and the probability of losing RAID 1 data is: ##EQU5##

Detailed Description Text (30):

Again, P(1,1.vertline.2) and P(6,1.vertline.2) are compared to determine which RAID level should be rebuilt first. The higher the probability, the higher the priority for that RAID level to be rebuilt first. Accordingly, for any Case 2 scenario where N>4, RAID 6 is the most vulnerable to potential data loss and thus will be prioritized to be rebuilt first for this case study.

Detailed Description Text (31):
Case 3: RAID 1 and RAID 6; Two Disks Failed, Mirror of Each Other

Detailed Description Text (32):
For Case 3, if the two failed disks are mirror of each other in RAID 1, we have already lost RAID 1 data on those two disks and, as such, rebuilding the lost data in RAID 1 is not possible. In this scenario, the only choice we have is to rebuild RAID 6 data. Thus, ##EQU6##

Detailed Description Text (33):
A comparison of P(1,1.vertline.2) and P(6,1.vertline.2) for Case 3 clearly indicates that the RAID 6 data is the most vulnerable to potential data loss because P(6,1.vertline.2) will always have the higher probability and thus the higher priority. In other words, we have no choice but to rebuild RAID 6 data. As such, in a Case 3 scenario, RAID 6 will always be rebuilt first.

Detailed Description Text (34):
Case 4: RAID 1 and RAID 6: Three Disks or More Failed

Detailed Description Text (35):
Finally for Case 4, if three disks or more have failed, then RAID 6 data is definitely lost, and the only data that may be rebuilt is in RAID 1 (if possible). In this case:

Detailed Description Text (36):
Clearly, for a Case 4 scenario, the only choice is to rebuild RAID 1 data when possible. As such, in a Case 4 scenario, RAID 1 will always be rebuilt first when possible.

Detailed Description Text (37):
Now, in summary, these Case examples have shown that for Case 3, we have no choice but to rebuild RAID 6, and for Case 4 the only choice is to rebuild RAID 1 when possible. On the other hand, decisions for Case 1 and Case 2 are dependent on MTTR, MTBF and N. As these numbers change (i.e.; faster repair time, faster rebuild time, longer time between failure, different number of disks supported in each redundancy group in the system) the strategy for rebuild priority may change accordingly. As an example, using generic numbers for demonstration purposes only, let us assume that: MTBF is 1,000,000 hours (1.0E+6 hours), MTTR is 24 hours, and N is 10;

Detailed Description Text (41):
This example shows again that for Case 1, we rebuild RAID 1 first. However, for Case 2 the rebuild strategy should focus on RAID 6 first.

Detailed Description Text (42):
Finally, consider the present invention rebuild prioritization system 35 in context of one more example of a hierarchical RAID array 15 that uses one redundancy group (group 1) having ten disks with RAID 1 and RAID 5 storage schemes (and not RAID 6), where each RAID level may be employed separately or jointly on any one or more of the disks in the array. The RAID 5 storage includes a stripe of data of a single block size on each disk. Each of nine of the disks holds actual user data in its respective block of the stripe. The tenth disk holds a block of data in the stripe containing redundant (parity) information. If a disk fails, the data on the failed disk can be reconstructed from the data on the remaining nine disks. The original and reconstructed data can then be re-written across the nine remaining good disks

in a new stripe (with one of those disks being designated as the parity disk). However, if any of the nine remaining disks fail before all of the RAID 5 data is rebuilt and re-written, then data will be lost.

Detailed Description Text (43):
For the RAID 1 storage of this example, data stored on a first of the ten disks is mirrored only on a second disk in the array. If the first disk fails, then during rebuild from the second (mirror) disk, data will only be lost if the second disk fails. Other disks could fail and the array would not lose data in RAID 1, but would lose data in RAID 5. Therefore, in this example of RAID 1 and RAID 5 hierarchical storage, in the event of a disk failure and during a rebuild, the RAID 5 storage has a greater probability of data loss than does the RAID 1 storage. Thus, the rebuild prioritization system 35 will cause RAID 5 data to be rebuilt first. The probabilities for this example are as follows: ##EQU7##

Detailed Description Text (44):
Referring now to FIG. 2 and FIG. 3, these flow charts depict a preferred method of the present invention and will be discussed in reference with FIG. 1. FIG. 2 depicts a preferred method from an overall, high level perspective, as embodied in two independently executing processes associated with rebuild prioritization system 35. Namely, when a disk failure is detected 105 in a hierarchical RAID data storage system 10, an interrupt is sent 110, 115 to initiate rebuild prioritization processing. Upon interrupt initiation 115, the present invention rebuild prioritization system 35 identifies 120 the most vulnerable RAID level in the array 15 relative to potential data loss with respect to the possibility of any further disk failure(s) during rebuild. Subsequently, upon identifying the most vulnerable RAID level, the rebuild prioritization system 35 communicates 125 with RAID management system 30 to enable, cause, or control RAID management system 30 to rebuild data of the failed disk by rebuilding the identified most vulnerable RAID level first. In this manner, importantly, minimizing the probability of data loss to the array 15 may be achieved. Additionally, the array's overall availability may be increased.

Detailed Description Text (45):
It should be noted here that if a next disk failure detection occurs 105 during a rebuild (i.e., during an already existing disk failure), the present invention rebuild prioritization system 35 again sends an interrupt 110, 115 to recalculate the rebuild prioritization 120 and enable or control a newly prioritized rebuild 125. In response, RAID management system 30 acknowledges any newly calculated rebuild prioritization determined, holds any currently executing rebuild, and then initiates and completes rebuild based on the newly calculated information. In a preferred embodiment, this rebuild prioritization processing is interrupt driven 110, 115 in order to identify the most vulnerable RAID level at any point in time and given any set of system changes or circumstances, even during an executing prioritization determination or data rebuild. For example, if a rebuild prioritization is occurring in response to a first disk failure in redundancy group 60 (FIG. 1), and a next disk failure actually occurs either in the same redundancy group or in a separate redundancy group, then an interrupt signal 115 is sent to re-initiate prioritization 120 and to identify a newly calculated most vulnerable RAID level and to enable the rebuild to occur based on the newly calculated most vulnerable RAID level.

Detailed Description Text (46):
FIG. 3 depicts further details of step 120 of FIG. 2. To identify the most vulnerable RAID level 120, characteristics associated with the storage system 10, 15 and storage devices 20 in the array are considered, including number of storage devices in each redundancy group, number of device failures, mean time between failure, mean time to rebuild, and failure dependencies. Failure dependencies include the location of the failure(s) with respect to redundancy group(s), which redundancy scheme(s) are affected by the failure(s) in the redundancy group(s), and

the relation of the failure(s) to each other within the affected redundancy scheme (s).

Detailed Description Text (47):
The first step is to identify 150 any redundancy group(s) 60, 65, 70 or 75 in the array. Next, 155, the RAID level(s) employed by each redundancy group is/are identified. For example, are RAID 1 and RAID 6 being used in a single redundancy group, or only RAID 1 in one redundancy group and only RAID 5 in a separate redundancy group, or only RAID 1 in separate redundancy groups, or some other combination of RAID levels and redundancy groups?

Detailed Description Text (48):
Next, storage device and system variables are identified 160 for each redundancy group and RAID level. These variables include, for example, the number of disks (N) in the stripe set or redundancy group(s), mean time between failure (MTBF), mean time to rebuild (MTTR), and the relation of the failure(s) to each other within the affected redundancy scheme(s).

Detailed Description Text (49):
Subsequently, 165, the probability of losing data (with respect to the potential for one or more next storage device failures in the data storage system) is calculated for each RAID level in each redundancy group. See the discussion of FIG. 1 for more specific examples of calculating probabilities. Then, the calculated probabilities are compared 170. Finally, 175, the comparison dictates that the RAID level with the highest probability is the RAID level that has the greatest vulnerability for potential data loss (relative to other RAID levels in the given redundancy group and relative to other redundancy groups and their RAID levels) with respect to the potential for one or more next storage device failures during a rebuild of the currently failed disk.

Detailed Description Text (50):
It should be noted that in a preferred embodiment, the rebuild prioritization system 35 of the present invention includes hard coded instructions and data for identifying the most vulnerable RAID level. In other words, in any given initially manufactured storage system 10, the maximum number of disks 20 in each redundancy group are known in advance, and their MTBF and MTTR values are also known. Additionally, the hierarchical RAID levels implemented are also known and limited in advance. Thus, all possible disk failure scenarios are known and, as such, the most vulnerable RAID levels given the different possible disk failure scenarios are also known in advance (can be compared or calculated in advance based on conditions). Accordingly, in a preferred embodiment, upon detection of a need to rebuild after a disk failure, the rebuild prioritization system 35 simply executes branching instructions based on previously compared or calculated probabilities that meet the disk failure conditions to quickly identify and enable RAID management system 30 to perform a rebuild upon the most vulnerable RAID level first.

Detailed Description Text (51):
Alternatively, rebuild prioritization system 35 includes data, tables or structures that define and identify storage system 10 in such detail that a most vulnerable RAID level is detected dynamically. In other words, the number of disks 20 in the system 10 are dynamically determined during any array uptime, the types of disks and their MTBF values are stored in memory (such as in a read only memory) for dynamic look-up accessability, and the MTTR values and RAID levels currently employed are dynamically calculated or determined. With these factors being dynamically obtained, rebuild prioritization system 35 literally calculates on-the-fly the most vulnerable RAID level so that it can be prioritized and rebuilt first and reports the same to RAID management system 30.

Detailed Description Text (52):

Finally, what has been described above are preferred embodiments for a storage management and data rebuild prioritization system for a hierarchical RAID array. It will be obvious to one of ordinary skill in the art that the present invention is easily implemented utilizing any of a variety of hardware platforms and software tools existing in the art. While the present invention has been described by reference to specific embodiments, it will be apparent that other alternative embodiments and methods of implementation or modification may be employed without departing from the true spirit and scope of the invention.

Detailed Description Paragraph Equation (1):
P(x,y.vertline.z)=0, when z=0 (i.e., no disk failure in redundancy group y), or RAID x data in redundancy group y is already lost (i.e., cannot be prevented)

Detailed Description Paragraph Equation (3):
P(6,1.vertline.z)=0 for z.gtoreq.3 (due to the fact that data in RAID 6 is already lost, and rebuild is impossible)

CLAIMS:

2. The method of claim 1 wherein the data redundancy schemes include RAID levels.

4. The method of claim 1 wherein controlling the data rebuild includes prioritizing the redundancy schemes and enabling a rebuild of the most vulnerable data redundancy scheme prior to enabling a rebuild of any other data redundancy scheme in the storage system.

13. A hierarchical data storage system, comprising: (a) data redundancy schemes; (b) failure detection apparatus configured to detect a failure of a storage device or devices in the storage system; and, (c) rebuild prioritization apparatus configured to enable a rebuild of data based on a most vulnerable data redundancy scheme in the storage system, wherein the most vulnerable data redundancy scheme is defined as a data redundancy scheme having a highest probability of losing data that can be prevented by a rebuild with respect to the potential for one or more next storage device failures in the data storage system.

14. The data storage system of claim 13 wherein the data redundancy schemes include RAID levels.

16. The data storage system of claim 13 wherein the rebuild prioritization apparatus is configured to enable a rebuild of the most vulnerable data redundancy scheme prior to enabling a rebuild of any other data redundancy scheme in the storage system.

17. The data storage system of claim 13 wherein the rebuild prioritization apparatus is configured to enable a rebuild of data based on a most vulnerable data redundancy scheme by considering characteristics associated with the data storage system, including at least a subset of the storage devices.

22. A hierarchical data storage system, comprising: (a). data redundancy schemes; (b) failure detection apparatus configured to detect a failure of a storage device or devices in the storage system; and, (c) rebuild prioritization apparatus configured to enable a rebuild of data based on a most vulnerable data redundancy scheme in the storage system by considering characteristics associated with the data storage system, including at least a subset of the storage devices.

First Hit Fwd Refs

End of Result Set

Generate Collection Print

L15: Entry 2 of 2 File: USPT Feb 4, 2003

DOCUMENT-IDENTIFIER: US 6516425 B1
TITLE: Raid rebuild using most vulnerable data redundancy scheme first

Brief Summary Text (2):
This invention relates in general to computer mass storage systems and, more particularly, to prioritizing data rebuild in the event of a disk failure in a hierarchical system utilizing a Redundant Array of Independent Disks (RAID).

Brief Summary Text (4):
Conventional disk array data storage systems have multiple disk storage devices that are arranged and coordinated to form a single mass storage system. A Redundant Array of Independent Disks (RAID) system is an organization of data in an array of mass data storage devices, such as hard disk drives, to achieve varying levels of data availability and system performance. Data availability refers to the ability of the RAID system to read and write data in the array of data storage devices even in the event of a failure of one or more of the individual data storage devices or components in the array. A measurement of system performance is the rate at which data can be sent to or received from the RAID system.

Brief Summary Text (6):
RAID systems typically designate part of the physical storage capacity in the array to store redundant data, either mirror or parity. The redundant information enables regeneration of user data in the event that one or more of the array's member disks, components, or the access paths to the disk(s) fail. Typically, the disks are divided into equally sized address areas referred to as "blocks." A set of blocks that has the same unit address ranges from each disk is referred to as a "stripe" or "stripe set." A set (or subset) of disks in the array over which a stripe or stripe set spans is referred to as a redundancy group. Traditionally, RAID arrays employ one or more redundancy groups and a single redundancy scheme for each redundancy group, although the schemes may vary among the redundancy groups. However, as will be discussed subsequently herein, hierarchical RAID arrays employ one or more redundancy schemes (i.e., RAID levels) for each redundancy group in an array.

Brief Summary Text (7):
From a data management and data redundancy perspective, RAID levels are typically characterized as one of six architectures, or redundancy schemes, enumerated as RAID levels 1-6. Although other RAID levels exist, levels 1-6 are the most commonly used and will be discussed herein with respect to the present invention. However, it should be noted that the present invention is applicable to any RAID level or data redundancy scheme.

Brief Summary Text (8):
The use of disk mirroring is referred to as RAID Level 1, where original data is stored on one set of disks and a duplicate copy of the data is kept on separate disks. The use of parity checking is referred to as RAID Levels 2, 3, 4, 5, and 6. In general, although RAID 1 provides higher data reliability and may provide the best small-write input/output (I/O) performance over RAID Levels 2, 3, 4 and 5, it

uses the most storage space because all data is duplicated. In contrast, RAID Levels 2-5 provide a lesser amount of data reliability (relative to RAID 1) and, typically, reduced small-write performance. However, they don't consume as much disk space as a RAID 1 technique because data is not duplicated but rather interleaved and parity checked across the disk array in a stripe set. A parity stripe set interleaves data and redundant (parity) data on multiple member disks. The parity stripe set presents a single virtual disk whose user data capacity is approximately the sum of the capacities of its members, less the storage used for holding the parity (redundant) data of the user data. For RAID levels 3-5, parity is commonly calculated using a bit by bit Exclusive OR function of corresponding data chunks in a stripe set from all of the data disks. This corresponds to a one equation, one unknown, sum of products calculation. The mirror set in a RAID 1 architecture presents a single virtual disk whose user data capacity is the sum of the capacity of one-half of its members, the other half holding the mirrored (redundant) data of the user data.

Brief Summary Text (9):
In addition to RAID mirror level 1, RAID parity levels 4, 5 and 6 are of particular interest for the present discussion. Specifically, for example, a RAID 4 uses a stripe set or redundancy group and a single dedicated parity disk to store redundant information about the data existing on the other data disks in the stripe set or redundancy group. Segments of data from each virtual disk sector are distributed across corresponding sectors of all but one of the stripe set members (i.e., the parity disk), and the parity of the distributed segments is written in the corresponding sector of the parity disk.

Brief Summary Text (10):
Because a RAID 4 system stores all parity blocks on a single unit in the stripe set, the single unit containing the parity blocks is accessed disproportionately relative to the other data storage devices in the stripe set or redundancy group. To eliminate the resulting constriction of data flow in a RAID 4 system, a RAID 5 architecture distributes the parity blocks across all of the data storage devices in the stripe set or redundancy group. Typically in a RAID 5 system, a set of N+1 data storage devices forms the stripe set or redundancy group. Each stripe has N blocks of data and one block of parity data. The block of parity data is stored in one of the N+1 data storage devices. The parity blocks corresponding to the remaining stripes of the stripe set or redundancy group are stored across the data storage devices within the stripe set or redundancy group. For example, in a RAID 5 system using five data storage devices in a given stripe set or redundancy group, the parity block for the first stripe of blocks may be written to the fifth device; the parity block for the second stripe of blocks may be written to the fourth device; the parity block for the third stripe of blocks may be written to the third device; etc. Typically, the location of the parity block for succeeding blocks shifts to the succeeding logical device in the stripe set or redundancy group, although other patterns may be used.

Brief Summary Text (11):
A RAID 6 architecture is similar to RAID 4 and 5 in that data is striped, but is dissimilar in that it utilizes two independent and distinct parity values for the original data, referred to here as P & Q. The P parity is commonly calculated using a bit by bit Exclusive OR function of corresponding data chunks in a stripe from all of the data disks. This corresponds to a one equation, one unknown, sum of products calculation. On the other hand, the Q parity is calculated linearly independent of P, but again using a different algorithm for sum of products calculation. As a result, each parity value is calculated using an independent algorithm and each is stored on a separate disk in the stripe set or redundancy group. Consequently, a RAID 6 system can rebuild data (assuming rebuild space is available) even in the event of a failure of two separate disks within a stripe set or redundancy group, whereas a RAID 5 system can rebuild data only in the event of no more than a single disk failure within a stripe set or redundancy group.

Brief Summary Text (12):
Similar to RAID 5, a RAID 6 architecture distributes the two parity blocks across all of the data storage devices in the stripe set or redundancy group. Thus, in a stripe set or redundancy group of N+2 data storage devices, each stripe has N blocks of data and two blocks of independent parity data. One of the blocks of parity data is stored in one of the N+2 data storage devices, and the other of the blocks of parity data is stored in another of the N+2 data storage devices. The parity blocks corresponding to the remaining stripes of the stripe set or redundancy group are stored across the data storage devices within the stripe set or redundancy group. For example, in a RAID 6 system using five data storage devices in a given stripe set or redundancy group, the parity blocks for the first stripe of blocks may be written to the fourth and fifth devices; the parity blocks for the second stripe of blocks may be written to the third and fourth devices; the parity blocks for the third stripe of blocks may be written to the second and third devices; etc. Typically, again, the location of the parity blocks for succeeding blocks shifts to the succeeding logical device in the stripe set or redundancy group, although other patterns may be used.

Brief Summary Text (13):
More information detailing the architecture and performance of RAID systems can be found in The RAID Book: A Source Book for RAID Technology, by the RAID Advisory Board, published Jun. 9, 1993, the disclosure of which is incorporated in full herein by reference. Additionally, a background discussion of RAID systems, and various ways to logically partition RAID systems, is found in U.S. Pat. No. 5,519,844 to David C. Stallmo, entitled "Logical Partitioning of a Redundant Array Storage System," incorporated in full herein by reference.

Brief Summary Text (14):
A hierarchical data storage system permits data to be stored according to one or more different techniques, such as different redundancy schemes, RAID levels, redundancy groups, or any combination of these. For example, in a hierarchical RAID system, data can be stored in one or more redundancy groups, and for each redundancy group, according to one or more RAID architectures (levels) to afford tradeoffs between the advantages and disadvantages of the redundancy techniques. For purposes of this disclosure, a hierarchical system includes: (1) one that automatically or dynamically migrates data between redundancy schemes (i.e., different RAID levels) and/or redundancy groups for optimum system tuning and performance, and (2) one that requires user input to configure redundancy schemes or groups, such as one that requires a user to designate a given RAID level with an identified set or subset of disks in a system array.

Brief Summary Text (15):
U.S. Pat. No. 5,392,244 to Jacobson et al., entitled "Memory Systems with Data Storage Redundancy Management", incorporated in full herein by reference, describes a hierarchical RAID system that enables data to be migrated from one RAID type to another RAID type as data storage conditions and space demands change. This patent describes a multi-level RAID architecture in which physical storage space is mapped into a RAID-level virtual storage space having mirror and parity RAID areas (e.g., RAID 1 and RAID 5). The RAID-level virtual storage space is then mapped into an application-level virtual storage space, which presents the storage space to the user as one large contiguously addressable space. During operation, as user storage demands change at the application-level virtual space, data can be migrated between the mirror and parity RAID areas at the RAID-level virtual space to accommodate the changes. For instance, data once stored according to mirror redundancy may be shifted and stored using parity redundancy, or vice versa.

Brief Summary Text (16):
With data migration, the administrator is afforded tremendous flexibility in defining operating conditions and establishing logical storage units (or LUNs). As

one example, this type of RAID system can initially store user data according to the optimum performing RAID 1 configuration. As the user data approaches and exceeds 50% of a stripe set or redundancy group array capacity, the system can then begin storing data according to both RAID 1 and RAID 5, and dynamically migrating data between RAID 1 and RAID 5 in a continuous manner as storage demands change. At any one time during operation, the data might be stored as RAID 1 or RAID 5 or both on all of the disks. The mix of RAID 1 and RAID 5 storage changes dynamically with the data I/O and storage capacity. This allows the system to dynamically optimize performance and available capacity versus an increasing amount of user data.

Brief Summary Text (17):
Clearly, each RAID level has characteristic cost-performance and cost-capacity ratios. Importantly, however, RAID systems maintain and manage redundant data to enable a recovery of the data in the event of a storage disk or component failure. To this regard, each RAID level also has a characteristic availability that determines the mean time to data loss as a function of the number of disks employed, i.e., different RAID levels provide different degrees of data protection. In the event of a disk or component failure, redundant data is retrieved from the operable portion of the system and used to regenerate or rebuild the original data that is lost due to the component or disk failure. Specifically, when a disk in a RAID redundancy group fails, the array attempts to rebuild data on the surviving disks of the redundancy group (assuming space is available) in such a way that after the rebuild is finished, the redundancy group can once again withstand a disk failure without data loss. Depending upon system design, the rebuild may be automated or may require user input. Design factors that affect rebuild include, for example, whether a spare disk is specified in the array, or whether the failed disk must be manually replaced by a user.

Brief Summary Text (18):
After detecting a disk or component failure and during a rebuild of data, regardless of rebuild design, the system remains subject to yet further disk or component failures in the same stripe set or redundancy group before the rebuild is complete. In any RAID system, this is significant because the vulnerability of data loss is dependent upon the RAID architecture (redundancy scheme) employed for that data.

Brief Summary Text (19):
For example, consider a hierarchical RAID array that uses ten disks in a single redundancy group with RAID 1 and RAID 5 storage schemes, where each RAID level may be employed separately or jointly on any one or more of the disks. The RAID 5 storage includes a stripe of data of a single block size on each disk. Each of nine disks holds actual user data in its respective block of the stripe. The tenth disk holds a block of data in the stripe containing redundant (parity) information. If a disk fails, the data on the failed disk can be reconstructed from the data on the remaining nine disks. The original and reconstructed data can then be re-written across the nine remaining good disks in a new stripe (with one of those disks being designated as the parity disk for the new stripe). However, if any of the nine remaining disks fail before all of the RAID 5 data is rebuilt and re-written, then data will be lost.

Brief Summary Text (20):
For the RAID 1 storage of this example, data stored on a first of the 10 disks is mirrored only on a second disk in the array. If the first disk fails, then during rebuild from the second mirror disk, data will only be lost if the second mirror disk fails. Other disks could fail and the array would not lose the RAID 1 data, but RAID 5 data would be lost.

Brief Summary Text (21):
Therefore, in this example of RAID 1 and RAID 5 hierarchical storage, in the event of a disk failure and during a rebuild, the RAID 5 storage is more vulnerable,

i.e., has a greater probability of data loss, than the RAID 1 storage. However, conventional hierarchical RAID systems rebuild data irrespective of the vulnerability of the RAID levels employed.

Brief Summary Text (22):
Accordingly, to minimize the probability of data loss during a rebuild in a hierarchical RAID system, there is a need to manage data recovery and rebuild that accounts for data availability characteristics of the hierarchical RAID levels employed.

Brief Summary Text (24):
According to principles of the present invention in a preferred embodiment, in a hierarchical data storage system employing data redundancy schemes, such as a RAID system, a method of managing data in response to a disk failure in the storage system includes prioritizing a data rebuild based on a most vulnerable data redundancy scheme identified in the storage system. Prioritizing the data rebuild includes enabling a rebuild of the most vulnerable data redundancy scheme prior to enabling a rebuild of any other data redundancy scheme in the system. The most vulnerable data redundancy scheme is determined by comparing a probability of losing data that can be prevented by a rebuild for each data redundancy scheme in the system with respect to the potential for one or more next disk failures in the data storage system. The probability of losing data for each data redundancy scheme is determined by considering characteristics associated with the storage system and storage devices in the array, such as number of storage devices, number of device failures, mean time between failure, mean time or calculated time to rebuild, and failure dependencies.

Detailed Description Text (2):
FIG. 1 is a block diagram of a hierarchical RAID data storage system 10 employing one embodiment of the present invention rebuild prioritization system 35 and method for prioritizing a data rebuild in the event of a disk failure, component failure or other system failure indicative of an inability to access certain data in the system. The data storage system 10 includes a disk array 15 having a plurality of storage disks 20, a disk array controller cluster 25 coupled to the disk array 15 to coordinate data transfer to and from the storage disks 20, and hierarchical RAID management system 30. The RAID management system 30 controls data storage and the hierarchical RAID levels (redundancy schemes) in the array 15, and controls the transferring of data among the various hierarchical redundancy schemes. Importantly, the RAID management system 30 includes a data rebuild prioritization system 35 according to the present invention. Specifically, in the event of a disk or component failure for any storage disk 20, the data rebuild prioritization system 35 minimizes potential data loss by prioritizing the rebuild according to the most vulnerable RAID level first.

Detailed Description Text (4):
The disk array controller cluster 25 is coupled to the disk array 15 via one or more interface buses 40, such as a small computer system interface (SCSI). The RAID management system 30 is operatively coupled to the disk array controller cluster 25 via an interface protocol 45. In the system shown, disk array controller cluster 25 is implemented as a dual controller consisting of disk array controller "A" 25A and disk array controller "B" 25B. Dual controllers 25A and 25B enhance reliability by providing continuous backup and redundancy in the event that one controller becomes inoperable. However, the present invention can be practiced with more than two controllers, a single controller or other architectures.

Detailed Description Text (5):
In a preferred embodiment, the RAID management system 30 and data rebuild prioritization system 35 are configured as firmware within a non-volatile random access memory (NVRAM) 25C, 25D of the disk array controller cluster 25. The RAID management system 30 and data rebuild prioritization system 35 include executable

instructions, routines, tables and/or other data and data structures necessary for managing the controller cluster 25 and array 15, including rebuilding data after a disk failure, and for prioritizing and/or managing the order or sequence of rebuild under the present invention as will be discussed more fully herein. The RAID management system 30 and data rebuild prioritization system 35 are shown as separate from the controller cluster 25 simply for clarity of depiction and descriptions purposes. Alternatively, the RAID management system 30 and data rebuild prioritization system 35 are embodied as a separate component (as shown), either as firmware or software, or configured within a memory of a host computer 50. Alternatively, these systems are embodied in hardwired circuitry (such as an ASIC). The data storage system 10 is coupled to the host computer 50 via an I/O interface bus 55.

Detailed Description Text (6):
It should be noted here that this disclosure does not detail general RAID management techniques or actual data rebuild techniques in the event of a disk failure because these are already sufficiently well known and disclosed in the art. Rather, the present invention disclosure focuses on identifying implementations and methods for prioritizing and/or managing the order or sequence of rebuild of an array system employing multiple (hierarchical) data redundancy levels and/or redundancy groups. Additionally, although a RAID system is specifically shown and discussed, it will be understood that the present invention is applicable to any data redundancy schemes, techniques, groups or systems.

Detailed Description Text (7):
The hierarchical disk array 15 is characterized by different storage spaces, including its physical storage space and one or more virtual RAID level storage spaces. These views of storage are related through mapping techniques. Specifically, the physical storage space of the disk array 15 is mapped into a virtual storage space which delineates storage areas according to the various RAID data reliability levels (redundancy schemes). For example, the storage disks 20 in the disk array 15 can be conceptualized as being arranged into multiple redundancy groups 60, 65, 70, 75, each having a defined subset of the disks 20, with each redundancy group being configured into stripe sets having stripes 60A, 60B, 60C, 65A, 65B, 65C, 70A, 70B, 70C, and 75A, 75B, 75C. Four exemplary redundancy groups are shown 60, 65, 70, 75, each having stripes zero through "N", but any number of groups or stripes are feasible under the present invention.

Detailed Description Text (8):
At any given point in time, each stripe is allocated a specific RAID level redundancy scheme and each redundancy group includes one or more RAID levels in its stripe set. In a preferred embodiment, RAID management system 30 migrates the redundancy schemes among the stripes and redundancy groups, depending upon storage capacity, data reliability and performance needs. For example, at a given point in time, stripe 0 (60A) of redundancy group 60 may be designated as RAID level 1 mirror redundancy, and stripes 1 (60B) and N (60C) may be designated as RAID level 5 parity redundancy. On the other hand, an entire redundancy group 65 may be designated as RAID level 6, and the other redundancy groups 70 and 75 may be designated as any combination of single or multiple redundancy schemes, including RAID levels 2, 3, 4, 5 or 6. Importantly, as described, each of these RAID levels may be configured on the same or separate disks or any combination thereof, all within stripe sets or redundancy groups in the disk array 15.

Detailed Description Text (9):
The data storage system 10 includes a memory map store that provides for persistent storage of the virtual mapping information used to map the disk array 15. The memory map store is external to the disk array and, preferably, redundantly resident in a memory 25C and 25D of the disk array controller cluster 25. The memory mapping information can be continually or periodically updated by the controller cluster 25 or the RAID management system 30 as the various mapping

configurations among the different views change.

Detailed Description Text (10):
As indicated, the disk array 15 has multiple storage disk drive devices 20, and the management of data redundancy on the devices 20 is coordinated by the RAID management system 30. When viewed by the user or host application program, an application-level virtual view can represent a single large storage capacity indicative of the available storage space on the storage disks 20 within the array 15. As mentioned, in a preferred embodiment, the RAID management system 30 dynamically alters the configuration of the RAID areas over the physical storage space. As a result, the mapping of the RAID areas in a RAID-level virtual view onto the disks and the mapping of a front end virtual view to the RAID view are generally in a state of change. The memory map store in memories 25C and 25D maintains the current mapping information used by the RAID management system 30 to map the RAID areas onto the disks, as well as the information employed to map between the virtual views. As the RAID management system dynamically alters the RAID level mappings, it also updates the mapping information in the memory map store to reflect the alterations.

Detailed Description Text (11):
Now, importantly, in the event of any disk failure relative to any one or more of the disks 20, the data rebuild prioritization system 35 of the present invention is activated to prioritize redundancy schemes (i.e., RAID levels) in the array 15 and enable a rebuild of the data in such a manner that potential data loss is minimized. Specifically, in response to a disk failure or multiple disk failures, the present invention prioritizes the redundancy schemes in each of the redundancy groups to enable a storage rebuild of the most vulnerable RAID level first, the second-most vulnerable RAID level next, and so forth. A most vulnerable redundancy scheme (i.e., RAID level) is defined as a redundancy scheme having the highest probability for potential data loss that can be prevented by a rebuild in the array 15 (relative to any other RAID levels within its redundancy group and relative to any other RAID levels within any other redundancy group in the array, all with respect to the potential for one or more next storage device failures in the array). Where a same redundancy scheme (i.e., RAID level) is employed in two or more separate redundancy groups, then a most vulnerable redundancy scheme may be defined relative to a given redundancy group.

Detailed Description Text (12):
Advantageously, this prioritization technique is applied to the array 15 regardless of the RAID levels or other data redundancy levels employed or that may be employed. The vulnerability is measured through the probability of losing data that can be prevented by a rebuild given that a disk or multiple disks 20 has/have failed. In a preferred embodiment, the data rebuild prioritization system 35 does not actually rebuild the data that is subject to the failure but, rather, identifies and prioritizes the order of rebuild so that the RAID management system 30 may perform the rebuild.

Detailed Description Text (13):
Due to the existence of multiple RAID levels on the same disk or disks 20 in a hierarchical architecture, or due to the existence of one or more RAID levels on one or more redundancy groups, enabling a rebuild of one RAID level over another by rebuild prioritization system 35 after a disk failure occurs may reduce exposure to subsequent disk failures that may actually lead to a loss of data. In this context, the order in which RAID levels are rebuilt may be critical because the probability of losing data may be reduced.

Detailed Description Text (14):
The order or sequence of rebuild is very dependent on the actual RAID levels that are supported by the disk storage system 10 and the sequence in which a disk failure or disk failures occur. Due to the huge number of combinations of disks

failures, and the RAID levels that may co-exist on a given disk array 15 or set of disks 20, the following examples will demonstrate the rebuild order (prioritization) analysis and steps performed by the data rebuild prioritization system 35 according to a preferred embodiment.

Detailed Description Text (15):
First, assume that we have a set of N disks 20 in a single redundancy group (group 1) and, for simplicity, assume N is even. Additionally, assume that all of the N disks are configured in the following RAID levels: RAID 1, where a portion of a disk is mirrored to another disk, and RAID 6, where N-2 disks contain actual data, and two disks contain two independent parities (two extra redundancy information).

Detailed Description Text (16):
Again, it should be noted that these are not the only RAID levels that can be supported by this invention; but for the purpose of demonstration, we are limiting our example to the above RAID levels.

Detailed Description Text (17):
Next, let us define conditional probability P(x,y.vertline.z) as the probability of losing RAID x (or redundancy scheme) data in redundancy group y that can be prevented by rebuild, given that z disks in redundancy group y have been identified as failed. In mathematical terms

Detailed Description Text (18):
P(x,y.vertline.z)=Pr (losing RAID x data in redundancy group y that can be prevented by rebuild .vertline.z disks in redundancy group y have been identified as failed)

Detailed Description Text (20):
Because a rebuild by rebuild prioritization system 35 does not start until a disk or multiple disk failures are encountered, the above probabilities must be conditioned on disk failure. Thus, for a single redundancy group system (y=1): P (1,1.vertline.1): Probability of losing RAID 1 data only in redundancy group 1 that can be prevented by rebuild given that one disk has failed in redundancy group 1; and P(6,1.vertline.1): Probability of losing RAID 6 data only in redundancy group 1 that can be prevented by rebuild given that one disk has failed in redundancy group 1.

Detailed Description Text (21):
Also, let MTBF correspond to disk Mean Time Between Failure, and MTTR correspond to Mean Time To Rebuild. The MTBF and MTTR characteristics are determined by conventional methods, such as using statistical samples, or obtained from disk manufacturer specifications. Although MTTR is used for this discussion, it will be understood that a calculated time to rebuild is also feasible and may be substituted for MTTR. Additionally, it is understood that the MTTR is RAID level dependent and system configuration dependent. However, for ease of discussion purposes, let us assume that the MTTR (that may include disk replacement time if no on-line spare is present) for any RAID level is constant. Also, let us assume that disk failures are random (and follow exponential distribution). Then, in this context, the steady state probability (limiting probability) of a given disk to fail during the rebuild time is calculated by:

Detailed Description Text (22):
Given the forgoing, the following scenarios will now be analyzed: Case 1: One disk failed Case 2: Two disks failed, not mirror of each other in RAID 1 Case 3: Two disks failed, mirror of each other in RAID 1 Case 4: Three disks or more failed.

Detailed Description Text (23):
Case 1: RAID 1 and RAID 6; One Disk Failed

Detailed Description Text (24):
For Case 1, we have one disk failed, and if the next disk failure happens to be the mirror image of the failed disk in RAID 1 while the failed disk is still being rebuilt, then RAID 1 data loss will occur. It follows: ##EQU1##

Detailed Description Text (25):
On the other hand, if the next disk failure is not a mirror of the failed disk in RAID 1 (any disk, as long as it is not a mirror of the first failed disk, N-2 possible ways), then no data loss in RAID 6 will occur. It is only the failure of a third disk (any surviving disk, N-2 possible ways) that will cause the RAID 6 data loss. Thus, ##EQU2##

Detailed Description Text (26):
Consequently, P(1,1.vertline.1) and P(6,1.vertline.1) are compared to determine which RAID level should be prioritized to be rebuilt first. Clearly, the higher the probability, the higher the priority for that RAID level to be rebuilt first. Accordingly, for most Case 1 scenarios, RAID 1 is the most vulnerable to potential data loss and thus will be prioritized to be rebuilt first. Notably, an exception exists to rebuild RAID 6 first when N becomes large. For example, RAID 6 is rebuilt first for this case study when: ##EQU3##

Detailed Description Text (27):
Case 2: RAID 1 and RAID 6; Two Disks Failed, not Mirror of Each Other

Detailed Description Text (28):
For Case 2, if the two disks that have failed are not a mirror of each other in RAID 1, then a next disk failure (any surviving disk, N-2 possible ways) during the rebuild will cause a data loss in RAID 6. Thus: ##EQU4##

Detailed Description Text (29):
However, RAID 1 data can be lost too if it happens that the third disk failure is a mirror of one of the other two failed disks. The third disk must be a mirror of the first failed disk or a mirror of the second failed disk in RAID 1 (there are two possible ways), and the probability of losing RAID 1 data is: ##EQU5##

Detailed Description Text (30):
Again, P(1,1.vertline.2) and P(6,1.vertline.2) are compared to determine which RAID level should be rebuilt first. The higher the probability, the higher the priority for that RAID level to be rebuilt first. Accordingly, for any Case 2 scenario where N>4, RAID 6 is the most vulnerable to potential data loss and thus will be prioritized to be rebuilt first for this case study.

Detailed Description Text (31):
Case 3: RAID 1 and RAID 6; Two Disks Failed, Mirror of Each Other

Detailed Description Text (32):
For Case 3, if the two failed disks are mirror of each other in RAID 1, we have already lost RAID 1 data on those two disks and, as such, rebuilding the lost data in RAID 1 is not possible. In this scenario, the only choice we have is to rebuild RAID 6 data. Thus, ##EQU6##

Detailed Description Text (33):
A comparison of P(1,1.vertline.2) and P(6,1.vertline.2) for Case 3 clearly indicates that the RAID 6 data is the most vulnerable to potential data loss because P(6,1.vertline.2) will always have the higher probability and thus the higher priority. In other words, we have no choice but to rebuild RAID 6 data. As such, in a Case 3 scenario, RAID 6 will always be rebuilt first.

Detailed Description Text (34):
Case 4: RAID 1 and RAID 6: Three Disks or More Failed

Detailed Description Text (35):
Finally for Case 4, if three disks or more have failed, then RAID 6 data is definitely lost, and the only data that may be rebuilt is in RAID 1 (if possible). In this case:

Detailed Description Text (36):
Clearly, for a Case 4 scenario, the only choice is to rebuild RAID 1 data when possible. As such, in a Case 4 scenario, RAID 1 will always be rebuilt first when possible.

Detailed Description Text (37):
Now, in summary, these Case examples have shown that for Case 3, we have no choice but to rebuild RAID 6, and for Case 4 the only choice is to rebuild RAID 1 when possible. On the other hand, decisions for Case 1 and Case 2 are dependent on MTTR, MTBF and N. As these numbers change (i.e.; faster repair time, faster rebuild time, longer time between failure, different number of disks supported in each redundancy group in the system) the strategy for rebuild priority may change accordingly. As an example, using generic numbers for demonstration purposes only, let us assume that: MTBF is 1,000,000 hours (1.0E+6 hours), MTTR is 24 hours, and N is 10;

Detailed Description Text (41):
This example shows again that for Case 1, we rebuild RAID 1 first. However, for Case 2 the rebuild strategy should focus on RAID 6 first.

Detailed Description Text (42):
Finally, consider the present invention rebuild prioritization system 35 in context of one more example of a hierarchical RAID array 15 that uses one redundancy group (group 1) having ten disks with RAID 1 and RAID 5 storage schemes (and not RAID 6), where each RAID level may be employed separately or jointly on any one or more of the disks in the array. The RAID 5 storage includes a stripe of data of a single block size on each disk. Each of nine of the disks holds actual user data in its respective block of the stripe. The tenth disk holds a block of data in the stripe containing redundant (parity) information. If a disk fails, the data on the failed disk can be reconstructed from the data on the remaining nine disks. The original and reconstructed data can then be re-written across the nine remaining good disks in a new stripe (with one of those disks being designated as the parity disk). However, if any of the nine remaining disks fail before all of the RAID 5 data is rebuilt and re-written, then data will be lost.

Detailed Description Text (43):
For the RAID 1 storage of this example, data stored on a first of the ten disks is mirrored only on a second disk in the array. If the first disk fails, then during rebuild from the second (mirror) disk, data will only be lost if the second disk fails. Other disks could fail and the array would not lose data in RAID 1, but would lose data in RAID 5. Therefore, in this example of RAID 1 and RAID 5 hierarchical storage, in the event of a disk failure and during a rebuild, the RAID 5 storage has a greater probability of data loss than does the RAID 1 storage. Thus, the rebuild prioritization system 35 will cause RAID 5 data to be rebuilt first. The probabilities for this example are as follows: ##EQU7##

Detailed Description Text (44):
Referring now to FIG. 2 and FIG. 3, these flow charts depict a preferred method of the present invention and will be discussed in reference with FIG. 1. FIG. 2 depicts a preferred method from an overall, high level perspective, as embodied in two independently executing processes associated with rebuild prioritization system 35. Namely, when a disk failure is detected 105 in a hierarchical RAID data storage system 10, an interrupt is sent 110, 115 to initiate rebuild prioritization processing. Upon interrupt initiation 115, the present invention rebuild prioritization system 35 identifies 120 the most vulnerable RAID level in the array

15 relative to potential data loss with respect to the possibility of any further disk failure(s) during rebuild. Subsequently, upon identifying the most vulnerable RAID level, the rebuild prioritization system 35 communicates 125 with RAID management system 30 to enable, cause, or control RAID management system 30 to rebuild data of the failed disk by rebuilding the identified most vulnerable RAID level first. In this manner, importantly, minimizing the probability of data loss to the array 15 may be achieved. Additionally, the array's overall availability may be increased.

Detailed Description Text (45):
It should be noted here that if a next disk failure detection occurs 105 during a rebuild (i.e., during an already existing disk failure), the present invention rebuild prioritization system 35 again sends an interrupt 110, 115 to recalculate the rebuild prioritization 120 and enable or control a newly prioritized rebuild 125. In response, RAID management system 30 acknowledges any newly calculated rebuild prioritization determined, holds any currently executing rebuild, and then initiates and completes rebuild based on the newly calculated information. In a preferred embodiment, this rebuild prioritization processing is interrupt driven 110, 115 in order to identify the most vulnerable RAID level at any point in time and given any set of system changes or circumstances, even during an executing prioritization determination or data rebuild. For example, if a rebuild prioritization is occurring in response to a first disk failure in redundancy group 60 (FIG. 1), and a next disk failure actually occurs either in the same redundancy group or in a separate redundancy group, then an interrupt signal 115 is sent to re-initiate prioritization 120 and to identify a newly calculated most vulnerable RAID level and to enable the rebuild to occur based on the newly calculated most vulnerable RAID level.

Detailed Description Text (46):
FIG. 3 depicts further details of step 120 of FIG. 2. To identify the most vulnerable RAID level 120, characteristics associated with the storage system 10, 15 and storage devices 20 in the array are considered, including number of storage devices in each redundancy group, number of device failures, mean time between failure, mean time to rebuild, and failure dependencies. Failure dependencies include the location of the failure(s) with respect to redundancy group(s), which redundancy scheme(s) are affected by the failure(s) in the redundancy group(s), and the relation of the failure(s) to each other within the affected redundancy scheme (s).

Detailed Description Text (47):
The first step is to identify 150 any redundancy group(s) 60, 65, 70 or 75 in the array. Next, 155, the RAID level(s) employed by each redundancy group is/are identified. For example, are RAID 1 and RAID 6 being used in a single redundancy group, or only RAID 1 in one redundancy group and only RAID 5 in a separate redundancy group, or only RAID 1 in separate redundancy groups, or some other combination of RAID levels and redundancy groups?

Detailed Description Text (48):
Next, storage device and system variables are identified 160 for each redundancy group and RAID level. These variables include, for example, the number of disks (N) in the stripe set or redundancy group(s), mean time between failure (MTBF), mean time to rebuild (MTTR), and the relation of the failure(s) to each other within the affected redundancy scheme(s).

Detailed Description Text (49):
Subsequently, 165, the probability of losing data (with respect to the potential for one or more next storage device failures in the data storage system) is calculated for each RAID level in each redundancy group. See the discussion of FIG. 1 for more specific examples of calculating probabilities. Then, the calculated probabilities are compared 170. Finally, 175, the comparison dictates that the RAID

level with the highest probability is the RAID level that has the greatest vulnerability for potential data loss (relative to other RAID levels in the given redundancy group and relative to other redundancy groups and their RAID levels) with respect to the potential for one or more next storage device failures during a rebuild of the currently failed disk.

Detailed Description Text (50):
It should be noted that in a preferred embodiment, the rebuild prioritization system 35 of the present invention includes hard coded instructions and data for identifying the most vulnerable RAID level. In other words, in any given initially manufactured storage system 10, the maximum number of disks 20 in each redundancy group are known in advance, and their MTBF and MTTR values are also known. Additionally, the hierarchical RAID levels implemented are also known and limited in advance. Thus, all possible disk failure scenarios are known and, as such, the most vulnerable RAID levels given the different possible disk failure scenarios are also known in advance (can be compared or calculated in advance based on conditions). Accordingly, in a preferred embodiment, upon detection of a need to rebuild after a disk failure, the rebuild prioritization system 35 simply executes branching instructions based on previously compared or calculated probabilities that meet the disk failure conditions to quickly identify and enable RAID management system 30 to perform a rebuild upon the most vulnerable RAID level first.

Detailed Description Text (51):
Alternatively, rebuild prioritization system 35 includes data, tables or structures that define and identify storage system 10 in such detail that a most vulnerable RAID level is detected dynamically. In other words, the number of disks 20 in the system 10 are dynamically determined during any array uptime, the types of disks and their MTBF values are stored in memory (such as in a read only memory) for dynamic look-up accessability, and the MTTR values and RAID levels currently employed are dynamically calculated or determined. With these factors being dynamically obtained, rebuild prioritization system 35 literally calculates on-the-fly the most vulnerable RAID level so that it can be prioritized and rebuilt first and reports the same to RAID management system 30.

Detailed Description Text (52):
Finally, what has been described above are preferred embodiments for a storage management and data rebuild prioritization system for a hierarchical RAID array. It will be obvious to one of ordinary skill in the art that the present invention is easily implemented utilizing any of a variety of hardware platforms and software tools existing in the art. While the present invention has been described by reference to specific embodiments, it will be apparent that other alternative embodiments and methods of implementation or modification may be employed without departing from the true spirit and scope of the invention.

Detailed Description Paragraph Equation (1):
P(x,y.vertline.z)=0, when z=0 (i.e., no disk failure in redundancy group y), or RAID x data in redundancy group y is already lost (i.e., cannot be prevented)

Detailed Description Paragraph Equation (3):
P(6,1.vertline.z)=0 for z.gtoreq.3 (due to the fact that data in RAID 6 is already lost, and rebuild is impossible)

CLAIMS:

2. The method of claim 1 wherein the data redundancy schemes include RAID levels.

14. The data storage system of claim 13 wherein the data redundancy schemes include RAID levels.

h e b b g e e e f c e h e ge

First Hit Fwd Refs

Generate Collection Print

L15: Entry 1 of 2 File: USPT Apr 29, 2003

DOCUMENT-IDENTIFIER: US 6557039 B1
TITLE: System and method for managing information retrievals from distributed archives

Brief Summary Text (5):
The hardware typically incorporated in an electronic archive is comprised of a general purpose computer and storage devices (such as magnetic disks, optical disks and magnetic tape subsystems). The hardware is typically operated and accessed by software comprising an operating system, database management systems, hierarchical storage management software (HSM) and archive management software. There are at least four significant limitations associated with current long term archival systems. First, larger corporations will invariably require several geographically diverse heterogenous archival systems in order to support the various operations of the corporation throughout the country and the world. For example, The corporation's research and development facility in London England has a separate archival system from the archival system for one of the corporation's manufacturing sites in Dallas Tex. Even if each of the archive facilities has a heterogeneous archival (e.g., a database manager) the hardware and the software comprising the archival at the two sites is invariably provided by two different vendors whose proprietary product are not interoperable (i.e., the software at the London site cannot be used to access the information stored at the Dallas site).

Detailed Description Text (3):
Schematically included in each of the archives 100-106 are the physical storage devices 110, the software 112 for accessing the physical devices 110, the site specific software 114 for controlling access to archived information, and site specific messaging system 116 for communication with a site. Typical storage devices 102 include Direct Access Storage Devices (DASD), optical storage devices and magnetic tape devices. These storage devices are typically configured in a hierarchical manner such that information that is more recent or that is more often accessed is stored on devices with the quickest access time, for example DASD. Using conventional archiving techniques, as electronic information "ages", it is migrated for archival purposes from DASD to devices with a slower access time such as optical disks or magnetic tape. Optical disks and tape provide a cost effective means for the storage of large quantities of electronic information. Tapes are typically stored and accessed through tape silos while a large quantity of optical disks are stored and accessed from one or more jukeboxes. Some specific examples of storage devices 110 include IBM and EMC magnetic disks, STK magnetic tape silos, Boxhill RAID magnetic disks, and Hewlett Packard magneto-optical jukeboxes.

Detailed Description Text (29):
The Priority Administration section 250 allows the manual intervention to change the priority number (01-99) for an individual request or a group of requests. This function allows dynamic priority re-assignment during periods where heavy request volumes are creating request backlogs.

L13: Entry 1 of 8 File: USPT Apr 13, 2004

DOCUMENT-IDENTIFIER: US 6721490 B1
TITLE: Hierarchical storage scheme and data playback scheme for enabling random access to realtime stream data

Abstract Text (1):
A hierarchical memory scheme capable of improving a hit rate for the segment containing the random access point rather than improving the overall hit rate of the cache, and a data playback scheme capable of automatically detecting positions that are potentially used as playback start indexes by the user and attaching indexes, are disclosed. The hierarchical storage device stores random access point segment information from which a possibility for each segment to contain a point that can potentially be random accessed in future can be estimated, and controls a selection of the selected segments to be stored in the cache storage device according to the random access point segment information. The data playback device records a plurality of playback start indexes, each playback start index being information regarding a playback position that is determined according to the user input which is recorded when the user input is in a prescribed pattern, and presents the plurality of playback start indexes to a user so as to urge the user to select a desired playback position.

Brief Summary Text (3):
The present invention relates to a hierarchical storage device (that can be used as a server side device) having a small capacity primary storage device and a large capacity secondary storage device which uses the primary storage device as a cache and its control method, and a data playback device (that can be used as a client side device) for playbacking data of video, audio, etc., stored in a storage device according to a user operation and its control method.

Brief Summary Text (7):
For these reasons, there has been a proposition of a hierarchical storage device which incorporates the generally known concept of hierarchical memory by using a hard disk or the like with a high transfer performance and a good random access handling as a primary memory while using the above described large capacity storage device as a secondary storage device.

Brief Summary Text (8):
In general, in such a hierarchical storage device, the primary storage device functions as a cache or a prefetch buffer of data stored in the secondary storage device. In the following, the primary storage device will also be referred to as a cache, while the secondary storage device is also referred to as a library.

Brief Summary Text (9):
In the hierarchical storage device, at a time of making an access to the stored data, whether the necessary data are stored in the cache or not is checked first. If the data are stored in the cache, the data on the cache are accessed. If the data are not stored in the cache, the data on the library are accessed while the accessed data are copied into the cache.

Brief Summary Text (28):
According to one aspect of the present invention there is provided a hierarchical

storage device, comprising: a library storage device configured to store realtime stream data in units of segments subdividing each realtime stream data; a cache storage device configured to store selected segments among the segments stored in the library storage device; a memory unit configured to store random access point segment information from which a possibility for each segment to contain a point that can potentially be random accessed in future can be estimated; and a control unit configured to control a selection of the selected segments to be stored in the cache storage device according to the random access point segment information stored in the memory unit.

Brief Summary Text (29):
According to another aspect of the present invention there is provided a method for controlling a hierarchical storage device formed by a library storage device storing realtime stream data in units of segments subdividing each realtime stream data and a cache storage device storing selected segments among the segments stored in the library storage device, the method comprising the steps of: storing random access point segment information from which a possibility for each segment to contain a point that can potentially be random accessed in future can be estimated; and controlling a selection of the selected segments to be stored in the cache storage device according to the random access point segment information stored by the storing step.

Drawing Description Text (2):
FIG. 1 is a block diagram showing an exemplary configuration of a hierarchical storage device according to the first embodiment of the present invention.

Drawing Description Text (3):
FIGS. 2A, 2B and 2C are diagrams for explaining exemplary conditions for RAP candidate detection that can be used in the hierarchical storage device of FIG. 1.

Drawing Description Text (4):
FIGS. 3A, 3B and 3C are diagrams showing exemplary configurations of a RAP segment information memory unit in the hierarchical storage device of FIG. 1.

Drawing Description Text (5):
FIG. 4 is a diagram showing an exemplary data structure for managing information necessary at a discarding segment selection unit in the hierarchical storage device of FIG. 1.

Drawing Description Text (6):
FIG. 5 is a flow chart showing an exemplary operation for selecting a segment list corresponding to a category in which a segment is to be contained in the hierarchical storage device of FIG. 1.

Drawing Description Text (7):
FIG. 6 is a flow chart showing an exemplary operation for updating a data structure at a discarding segment selection unit in the hierarchical storage device of FIG. 1 in the case where data is closed.

Drawing Description Text (8):
FIG. 7 is a flow chart showing an exemplary operation for updating a data structure at a discarding segment selection unit in the hierarchical storage device of FIG. 1 in the case where data is opened.

Drawing Description Text (9):
FIG. 8 is a flow chart showing an exemplary operation for updating a data structure at a discarding segment selection unit in the hierarchical storage device of FIG. 1 in the case where a segment is newly stored in a cache.

Drawing Description Text (10):

FIG. 9 is a flow chart showing an exemplary operation or updating a data structure at a discarding segment election unit in the hierarchical storage device of FIG. 1 in the case where a segment is discarded from a cache.

Drawing Description Text (11):
FIG. 10 is a flow chart showing an exemplary operation or updating a data structure at a discarding segment selection unit in the hierarchical storage device of FIG. 1 in the case where a RAP segment information is updated.

Drawing Description Text (12):
FIG. 11 is a flow chart showing an exemplary operation for sequentially specifying categories in order to select a candidate for discarding from a cache in the hierarchical storage device of FIG. 1.

Drawing Description Text (13):
FIG. 12 is a flow chart showing an exemplary operation for searching a segment within a specified category in order to select a candidate for discarding from a cache in the hierarchical storage device of FIG. 1.

Drawing Description Text (14):
FIG. 13 is a flow chart showing an exemplary operation for updating a data structure at a storing segment selection unit in the hierarchical storage device of FIG. 1 in the case where data is opened.

Drawing Description Text (15):
FIG. 14 is a flow chart showing an exemplary operation for updating a data structure at a storing segment selection unit in the hierarchical storage device of FIG. 1 in the case where data is closed.

Drawing Description Text (16):
FIG. 15 is a flow chart showing an exemplary operation for updating a data structure at a storing segment selection unit in the hierarchical storage device of FIG. 1 in the case where a RAP segment information is updated.

Drawing Description Text (17):
FIG. 16 is a flow chart showing an exemplary operation for judging whether or not to store a prefetch candidate into a cache in the hierarchical storage device of FIG. 1.

Detailed Description Text (5):
FIG. 1 shows an overall configuration of a hierarchical storage device according to the first embodiment. In FIG. 1 an input/output unit 11 is a unit that functions as an interface with respect to the external of this hierarchical storage device. In the case where this hierarchical storage device is configured as an instrument that operates by being connected to a host computer, the input/output unit 11 contains a connection interface with respect to the host computer and its control software, and carries out transmission and reception of requests and data with respect to the host computer. In the case where this hierarchical storage device is configured as a computer that operates as a file server, the input/output unit 11 contains a network interface and its control software, and carries out transmission and reception of requests and data with respect to a client.

Detailed Description Text (6):
A secondary storage device 14 (which will also be referred to as a library hereafter) is formed by a magnetic tape device, an automatic tape changer for magnetic tapes, or an automatic disk changer for removable disks, so as to realize a large capacity inexpensively. The library stores all data to be stored in the hierarchical storage device of the first embodiment by dividing them into segments. Here, a segment is a unit of management which subdivides data into a size specific to the system. The size of a segment is determined in view of a total capacity and

transfer rates of the library and a cache storage device, a transfer rate at a time of utilizing data, etc.

Detailed Description Text (8):
The hierarchical storage device of FIG. 1 also include a request processing unit 12, a segment transfer management unit 13, a RAP (Random Access Point) candidate detection unit 16, a discarding segment selection unit 17 and a storing segment selection unit 18, which can be collectively provided as a control unit. The control unit controls segment transfer between the library and the cache, so as to realize improvement of the performance that is regarded important for the hierarchical storage device such as shortening of an average response time and improvement of a total transfer performance. In the first embodiment, a RAP segment information stored in a RAP segment information memory unit 19 is used as one of the criteria for judgement at a time of selecting a segment to be stored into the cache or a segment to be discarded from the cache.

Detailed Description Text (35):
In this hierarchical storage device, it suffices to have selected ones of mechanisms that can estimate the random access point more efficiently according to the types of requests that can be used or the type of use of the entire system including the client and the host computer, among the various mechanisms described above.

Detailed Description Text (36):
The RAP segment information to be used in this hierarchical storage device can be given in any of several forms described below, land storing in the RAP segment information memory unit 19, updating from the RAP candidate detection unit 16, and referring from the discarding/storing segment selection unit 17/18 will take forms suitable for the form of the RAP segment information used.

Detailed Description Text (46):
Next, the concrete example of the hierarchical storage device of the first embodiment realized as a server for handling a large capacity realtime stream such as video and audio will be described.

Detailed Description Text (47):
First, the system configuration in this concrete example will be described. The hierarchical storage device is connected with a client device through a network. The client device reads out the realtime stream from the hierarchical storage device through the network and playback them, or record the realtime stream into the hierarchical storage device through the network, according to operations of the user.

Detailed Description Text (49):
The client device issues a various types of requests to the hierarchical storage device according to the user operations. In the case of handling data as general files, general file operation requests such as "open", "seek", "read", "write", "close", etc., are provided. In the case of handling data as the realtime stream, when the user makes an operation such as playback or fast forward, the above described general file operation requests are appropriately combined according to that operation at the client device side and issued to the hierarchical storage device side. Else, a request corresponding to each user operation may be provided separately, and issued to the hierarchical storage device side according to the user operation.

Detailed Description Text (50):
Next, the configuration and the operation of the hierarchical storage device in this concrete example will be described. An automatic disk changer capable of storing a plurality of optical disks is used as the library storage device 14, while a hard disk device is used as the cache storage device 15. A network

interface and its control software are provided at the input/output unit 11.

Detailed Description Text (53):
Next, the basic operation at a time of transmitting data from the hierarchical storage device to the client device as the realtime stream in response to a request from the client will be briefly described.

Detailed Description Text (64):
Condition 4: In the case where a random access point registration request is provided as a request from the client device to the hierarchical storage device, at a timing where this registration request is received, the specified point is detected as the RAP candidate

Detailed Description Text (68):
For example, in the case where a random access point registration request from the client is provided, the registration requested point can be considered as having a higher possibility of becoming the random access point subsequently than points obtained by other conditions for estimating the RAP candidate at the hierarchical storage device side from the tendency of the seek request, so that the increment of the RAP counter in the case of updating the RAP counter according to the condition 4 can be made larger than the other conditions.

Detailed Description Text (105):
Also, when it is assumed that the discarding priority order is higher in an order of the above described categories (a), (b), (c) and (d), there is a possibility for the RAP segments to occupy most of the cache. When this happens, the storing/discarding of normal segments that are not RAP segments will occur frequently except for those segments in the keep state. When the frequency of the segment transfer between the library and the cache increases too much, a problem concerning the durability of mechanical parts of the library may arise. In the case where this problem arises, the discarding priority order can be changed to the order of (a), (c), (b) and (d) if a rate by which the cache is occupied by the RAP segments exceeds a prescribed threshold. In this way, in addition to reducing the response time at a time of the random access, it becomes possible to realize a better balance with respect to the frequency of the segment transfer between the library and the cache.

Detailed Description Text (158):
The data input/output unit 22 is a unit for handling a request with respect to the external storage device and read/write of data recorded in the external storage device through the data communication path. For example, in the case where this data playback device is used as a client that carries out read/write of data with respect to a storage device (which can be the hierarchical storage device of FIG. 1 or the other type of server device) that functions as a server through a network, the data input/output unit 22 is formed by a network interface and its control software, and carries out transmission and reception of request and data with respect to the server.

Detailed Description Text (178):
The detection of the seek points that can potentially be the playback start indexes in this data playback device as described above is based on the same principle as the detection of a top of the sequential accesses (playback start point) in the case where the sequential access (playback) amount is large (FIG. 2C) in the hierarchical storage device of FIG. 1. Namely, in the hierarchical storage device of FIG. 1, by updating the RAP segment information of a point that satisfies the condition as shown in FIG. 2C and controlling such that the probability for segments containing the RAP candidates to exist in the cache becomes high, the response time in the case where the user subsequently specifies the playback from arbitrary position in a middle of the contents is shortened at the server device side. On the other hand, in this data playback device, a point that satisfies the

condition (shown in FIG. 19) similar to that shown in FIG. 2C is recorded as a past record (playback start index list) at the client device side when the user carries out the initial search, so that it becomes possible to quickly call up the desired scene to be playbacked by utilizing that record subsequently.

Detailed Description Text (211):
The control unit 21 issues a seek request with respect to the server by setting the seek point pointed by the playback start index selected by the operator as argument, and then issues a data transfer request. The server moves the playback start position to the specified seek point, and starts the data transfer with respect to this data playback device starting from that position. At this point, if the server is the hierarchical storage device of FIG. 1 which has left or prefetched a segment containing that playback start position in the cache by detecting the RAP candidates according to the principle similar to that of the playback start index detection, the response time with respect to the data transfer request from this data playback device can be shortened.